TIMEX SINCLAIR™

Color Graphics

Nick Hampshire

FOR THE
TS 2048
AND
TS 2068
MICROCOMPUTERS

HAYDEN

TIMEX SINCLAIR™

Color Graphics

Nick Hampshire

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

Production Editor: TERRY DONOVAN
Art Director: JIM BERNARD
Printed and bound by: MAPLE-VAIL BOOK MANUFACTURING GROUP

Printed in the United States of America

1	2	3	4	5	6	7	8	9	PRINTING
83	84	85	86	87	88	89	90	91	YEAR

CONTENTS

AN OVERVIEW

The provision of low cost, high resolution colour graphics is probably one of the most exciting and challenging features of popular home computers such as the TS 2048 or 2068. With these features a whole new range of exciting applications are opened up for the adventurous programmer. Applications which involve the true visual display of concepts, ideas, and fantasies. In this book I hope to show you how to realise some of the colossal graphics display potential possessed by your machine.

To stimulate your imagination let's first look at some of the possibilities presented by a high resolution colour graphics computer. Perhaps the most obvious application is in simulations, and the most obvious use of simulations is in education. There is an old saying when trying to explain a concept, that a picture is worth a thousand words. This is particularly true in all science related subjects. Relationships can be shown between two or more mathematical functions displayed as curves on the screen, or a mathematical process such as differentiation can be shown graphically taking place. In chemistry three dimensional graphics can be used to show molecular structures and bonding. A chemical process can be displayed and the various reactions can be simulated by the computer.

Some of the best examples of simulations involving high resolution colour computer graphics come from physics. The teacher has the ability to easily display the concepts of mechanics, such as Newton's laws, the trajectory of a missile or planetary motion. Magnetic and electrostatic fields and their interrelationship can easily be displayed, as can the path of light through optical systems. In electronics the computer can be used to simulate a circuit and the high resolution graphics can be used to display the circuit on the screen.

Computer games — which are in the majority of cases just a special fun form of simulation — are obvious candidates for improvement by the use of high resolution colour graphics displays. Although the TS 2000 series still cannot match the incredible real time and very realistic displays found on many of the best arcade games, the quality of the home computer's graphics does allow for the programming of some fantastic display based games. In all these games programmes the graphics display is augmented by the sound generation ability of the TS 2000 series. The range of computer games is enormous, ranging from arcade games like Space Invaders or Packman to chess programmes with a high

quality display of a chess board and all the pieces, or the fantasy games like Adventure which can be endowed with some very interesting graphics.

Computer art is an application for high resolution colour computer graphics in which a growing number of people are becoming increasingly interested. The artist uses the graphics display as a canvas on which the picture or design is drawn either in a single colour or using all the colours available on the computer. The picture is created by using either a specially written program and an input data base to generate the displays, or a light pen or joystick to interactively paint the picture on the screen, much as one would by using a paint brush. Such displays could of course be either a static one off picture or an animated sequence. The generation of animated computer art displays is a subject of increasing interest to creators of cartoon films; this should be within the capabilities of the TS 2000 series. An example of such graphics was shown in the film 'Star Wars' in the scene where the rebel pilots are briefed on the workings of the 'Death Star'. Full length feature animated films generated by computer can be expected within the next year.

An important application for graphics simulation is using three dimensional graphics software to aid the designing of buildings or engineering structures. This is known as CAD, or Computer Aided Design, and although in commercial applications it is confined to very large fast computers it is quite possible to perform most of the CAD operations on machines such as the TS 2000 series. The designer builds up a model in the computer memory and, by using this data base, can view the structure from any angle or even go inside. Perspective, light and dark shading, surface texture and colour of solids can all be emulated by such software; some examples of routines to do these functions are given in the last section of this book. Another variation of this type of application is used in flight simulators, where the computer, by using a previously entered data base, creates a simulated display of a piece of terrain or an airfield as the person using the simulator would see it from any position in three dimensional space. In a flight simulator the position of viewing would depend on how the 'pilot' moved his controls. Simulated landings and take offs can thus give a visual feedback to the pilot through the use of such computer graphics.

COLOUR PLOTTING

COLOUR

An introduction to the colour commands on the TS 2000 series.

The computer screen is organised as 24 lines of 32 characters, and the character and background colour of each one of these 768 character spaces can be individually programmed to one of the eight possible colours which can be displayed by the computer. The two colours associated with each character space are the foreground or character colour, this is referred to as the INK colour, and the background colour or PAPER. In the normal power up mode the INK colour is black and the PAPER colour white.

There are eight different colours, including black and white, which can be displayed, they are as follows:

0 — black
1 — blue
2 — red
3 — purple or magenta
4 — green
5 — pale blue or cyan
6 — yellow
7 — white

These colours are produced on a colour TV by mixing just three primary colours — blue, red and green. Thus magenta, which is colour 3, is produced by mixing colours 1 and 2 — blue and red. Likewise colour 5, cyan, is a mix of colours 1 and 4, and colour 6, yellow is a mix of colours 4 and 2. From this you can see that the colour number is in fact the sum of the primary colours required to produce that colour thus white which is produced by having all three primary colours mixed has colour number 7 or colours 1 + 2 + 4. The number associated with each colour on the above list is important since it is used in the colour commands to designate that colour.

The INK command is used to set the character or foreground colour of characters subsequently displayed using PRINT commands starting at the current cursor position. The command:

```
INK 4 : PRINT "ink colour green"
```

will print the statement "ink colour green" on the screen starting at the current cursor position in green characters on the existing background colour (normally white) of the screen. To show the

range of colours try the following program:

```
10 FOR Q = TO 7
20 INK Q
30 PRINT "ink colour number"; Q
40 NEXT Q
```

The PAPER command is identical to the INK command except that it sets the background colour for the printed characters. Thus the command:

```
PAPER 4 : PRINT "paper colour is green"
```

will display the statement "paper colour is green" starting at the current cursor position and using the existing ink colour (normally black). The following short program shows the 64 different combinations of INK and PAPER colours which can be obtained using the two commands:

```
10 PRINT "01234567 ink colours"
20 FOR Q = 0 TO 7
30 FOR Z: PAPER Q
50 PRINT "*";
60 NEXT Z
70 PRINT "paper colour";Q
80 NEXT Q
```

Besides the foreground and background colours there is also the colour of the border around the screen display area. This border can have its colour set by use of the BORDER command followed by one of the eight colour code numbers. Thus the command:

```
BORDER 5
```

will set the screen border to a cyan colour.

The original ink or paper colours can be retained for a character by setting the colour value to 8. This means that the characters printed following the command are "transparent", with the previously defined colours on the screen being used to display the new characters. Thus if the command

```
PAPER 8
```

is executed then the paper colour will be left as currently displayed

on the text following the cursor. However, the ink colour will be that defined in the previous statement. Similarly the command:

INK 8

will leave the ink colour unchanged but the paper colour changed to that defined in the previous colour definition statement. Both INK 8 and PAPER 8 can be used together to leave all colours unchanged.

Contrast between some of the colours is very poor. For example, it is virtually impossible to read a character which has an ink colour of cyan and a paper colour of green. To overcome this and ensure enhanced character contrast there is an extra character code value. By using the colour code number 9 after either the INK or PAPER commands. This sets the colour used with either the defined INK or PAPER colour to a colour with the maximum contrast. Thus if the colour is dark (eg. black, blue, red or magenta) then the complementary colour will be made white or if light then the complementary colour will be black.

There are then a series of commands which can be used to control the way particular characters are displayed without actually altering the dot pattern or colours of each character space. The first three of these commands are BRIGHT, INVERSE and FLASH.

The BRIGHT command will display the background colour of the printed string following the BRIGHT statement with an enhanced brightness. This means that it will stand out in relation to other displayed strings which are used without the BRIGHT command. The number following the BRIGHT command determines whether it is turned on or off, a 0 and the "bright" is off and 1 the "bright" is on. The following is an example of a command using BRIGHT:

```
10 PRINT INK 0; PAPER 7; BRIGHT 1; "this is in bright mode"
20 PRINT INK 0; PAPER 7; BRIGHT 0; "the bright mode is turned off"
```

The INVERSE command simply reverses the foreground and background colours for the characters in the printed string after the INVERSE command. It does this without changing the dot pattern printed on the screen. To turn the INVERSE command on it should be followed by a 1, and to turn it off then it should be followed by a 0. The following is an example of the INVERSE command:

```
10 PRINT INK 0; PAPER 7; INVERSE 1; "characters are
inversed"
20 PRINT INK 0; PAPER 7; INVERSE 0; "characters returned
to normal"
```

The FLASH command is used to set a following character string to flash on and off between the normal screen display and the inverted display produced by the INVERSE command. The rate of flashing is about 3 times per second. This command like the previous two commands is very useful in drawing attention to a displayed statement or command. The following is an example of the FLASH command:

```
10 INPUT FLASH 1; INK 1; PAPER 7; "input data";N
```

This computer has a very useful overprinting command called OVER which allows one to create new characters by overprinting one or more characters over an existing character. The most obvious use of this command is to add an accent to a character. Normally when a character is displayed whatever is already written in that character space is obliterated, but in the OVER command the existing character is retained and the dots of the new character added. As with the previous commands following it with a 1 will turn it on and a 0 will turn it off. The following is an example of the OVER command:

```
10 OVER 1
20 PRINT "a"; CHR$8;"''";
```

note: the CHR$8 causes the cursor to back up one character space.

All the commands which control the attribute of a character can also be set using the character codes which represent the command thus the following commands and codes are identical:

CHR$ 16 — INK command
CHR$ 17 — PAPER command
CHR$ 18 — FLASH command
CHR$ 19 — BRIGHT command
CHR$ 20 — INVERSE command
CHR$ 21 — OVER command

ADVANCED COLOUR

Each one of the 768 character positions on the computer screen can be assigned two different colours, a foreground character colour and a background colour. These two colours can be selected from the eight different colours which can be displayed. To set the background colour for a character the PAPER command is used and for the character colour the INK command, both are followed by the required colour number. These colour values, or attributes as they are called in the Sinclair manual, are stored in a 768 byte section of memory stretching from location 22528 to 23296. In the eight bits of each byte in the attribute table are stored the ink and paper colour values plus two flags which indicate if the displayed character is in the normal bright or flashing mode. They are stored as follows:

7	6	5 4 3	2 1 0	bit number
FLASH	BRIGHT	PAPER	INK	

To set the INK colour for a character, a binary value between 0 and 7 is placed in the first three bits of the attribute memory byte corresponding to that character. Similarly to set the PAPER colour a binary value between 0 and 7 is placed in bits 3,4 and 5 of the appropriate attribute byte.

A characters colours can be set by using the POKE command to put the required colour values into the attribute RAM byte corresponding to the displayed character. It is much easier however, to use the INK and PAPER commands. A knowledge of how the colour values are stored is necessary if you wish to use the ATTR (line, col) command, this returns a value equal to the contents of the attribute memory byte at the assigned character coordinates.

Although there are only eight different colours, interesting coloured displays can be produced by a careful combination of foreground and background colours. The background PAPER colours can be used to give the main coloured display and the foreground INK colours can be used for the high resolution and text details of the display. An example of this kind of display is shown in the program MAP. The use of colour in high resolution graphics is unfortunately not very effective. The reason being that colours can only be defined on an 8 x 8 dot character resolution. The colour of an individual dot can thus not be changed without changing the

colour of all the other dots within the character space containing that dot. Similarly the background PAPER colour can only be defined on a character space resolution. One of the results of this limitation is that it is almost impossible to have two intersecting high resolution lines of different colours.

RANDOM COLOURS

DESCRIPTION

Background — PAPER — colours can be used to fill blocks of the screen with different colours thereby generating interesting effects. This program shows how the paper colour command can be used to generate colourful dynamically moving patterns. The display consists of a dynamically moving point at which are plotted squares of different colours, the movement of the point and the colour selection are random. The resulting display is a changing pattern of variously shaped different coloured blocks.

RUNNING THE PROGRAM

Since no parameters are input by the program, simply type RUN and watch the program display a constantly changing coloured pattern.

PROGRAM STRUCTURE

50	draw border around screen using subroutine at 400
110	initialise random seed
120	set starting point on screen
160-170	set random variables for colour and number of characters of same colour
210	set random variable for movement direction
220-260	move in one of four directions
280-310	check character position is within screen boundary
350-360	plot coloured square using a 'space' character
400-460	border drawing subroutine

```
   1 REM RANDOM COLOURS
   2 REM ************************
   3 REM
  10 REM the routine generates a
 dynamically moving colour displ
ay
  20 REM
  30 REM
  40 REM draw border around disp
lay
  50 GO SUB 400
  60 REM
 100 REM set constants
 110 RANDOMIZE
 120 LET a=10: LET b=20
 130 REM
 140 REM randomize colour and nu
mber of characters plotted varia
bles
 150 REM
 160 LET c=INT (RND*9)
 170 LET n=INT (RND*20)
 180 REM
 190 REM main character plotting
 routine
 200 REM
 210 FOR x=0 TO n
 220 LET d=INT (RND*4)
 230 IF d=0 THEN LET a=a+1
 240 IF d=1 THEN LET a=a-1
 250 IF d=2 THEN LET b=b+1
 260 IF d=3 THEN LET b=b-1
 265 REM
 270 REM within bounds?
 275 REM
 280 IF b>=30 THEN LET b=30
 290 IF b<=1 THEN LET b=1
 300 IF a>=20 THEN LET a=20
 310 IF a<=1 THEN LET a=1
 320 REM
 330 REM plot coloured character
 340 REM
 350 PAPER c
 360 PRINT AT a,b;" "
 370 NEXT x
 380 GO TO 160
 400 REM border drawing subrouti
ne
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

MAP

DESCRIPTION

Background colours can very effectively be used to fill blocks of the screen with different colours to define outline shapes. High resolution or character plotting can then be used to put details on the outline. This program shows how this can be done and to illustrate the technique draws a map of North America with appropriate text legends. The background colours are set by POKEing the correct colour value into the colour attribute memory (this is 704 bytes long and starts at location 22528). In this program only two outline colours are used and for this reason the plotting is divided into two sections, one for each colour. The display is built up from lines or single characters of colour. The data for each line displayed is stored as data statements and consists of sets of three values — line number, column number and number of characters from that position to be plotted continuously on the line. If the display was to be plotted in many different colours then an extra colour parameter should be added to the data tables.

RUNNING THE PROGRAM

Since no parameters are input by the program simply type RUN and watch the program display a map of North America on the screen using different colours for each country.

PROGRAM STRUCTURE

100-160	fill the screen with cyan colour to act as a background to the display
180	jump to border drawing subroutine at 1000
200-280	plot the map of USA in green using data from table lines 310-370
300-370	data table for drawing map of USA, note that the data is stored as a sequence of three values: line, column and length of block
500-570	plot the map of Mexico and Canada in white using data from the table in lines 600-700
600-700	data table for plotting Mexico and Canada
900-960	put legends on map — note: make sure that the paper colour for the text or high resolution is identical to that of the background colour already plotted
1000-1060	border drawing subroutine

```
   1 REM MAP
   2 REM ************************
   3 REM
  10 REM this program draws a co
loured map of North America
  20 REM
  30 REM
 100 REM set map background colo
ur as cyan
 110 REM
 130 PAPER 5
 140 FOR q=1 TO 22
 150 PRINT "
                                "
 160 NEXT q
 165 REM
 170 REM draw border around scre
en
 175 REM
 180 GO SUB 1000
 190 REM
 200 REM draw the USA in green
 205 REM
 210 READ r,s,l
 220 IF r=100 THEN GO TO 500
 230 LET p=22528+(r*32)+s
 240 FOR q=1 TO l
 250 POKE p+q,32
 260 NEXT q
 270 REM
 280 GO TO 210
 290 REM
 300 REM data for plotting USA
 305 REM
 310 DATA 5,26,1,6,5,4,6,25,3
 320 DATA 7,5,14,7,24,3,8,4,16,8
,23,3
 330 DATA 9,4,17,9,22,4,10,4,21
 340 DATA 11,4,20,12,4,20,13,5,1
9,14,5,19
 350 DATA 15,6,17,16,10,13,17,11
,8,17,21,2
 360 DATA 18,12,3,18,22,2,19,23,
1
 370 DATA 100,100,100
 495 REM
 500 REM draw Canada and Mexico
in white
 505 REM
 510 READ r,s,l
 520 IF r=100 THEN GO TO 900
 530 LET p=22528+(r*32)+s
 540 FOR q=1 TO l
 550 POKE p+q,56
 560 NEXT q
 570 GO TO 510
 595 REM
 600 REM data for Canada and Mex
ico
```

```
 604 REM
 605 DATA 0,2,24,1,3,24
 610 DATA 2,4,24,3,4,23,4,4,22,5
,5,21,6,9,16,7,19,5,8,20,3,9,21,
1
 620 DATA 16,6,4,17,6,1,17,8,3,1
8,7,1,18,9,3,19,9,6,20,9,6
 630 DATA 21,9,7
 700 DATA 100,100,100
 895 REM
 900 REM put names on map
 905 REM
 910 PAPER 7
 920 PRINT AT 3,12;"CANADA"
 930 PRINT AT 20,10;"MEXICO"
 940 PAPER 4
 950 PRINT AT 13,14;"USA"
 960 STOP
 995 REM
1000 REM draw border subroutine
1005 REM
1010 PLOT 0,0
1020 DRAW 255,0
1030 DRAW 0,175
1040 DRAW -255,0
1050 DRAW 0,-175
1060 RETURN
```

RAINBOW

DESCRIPTION

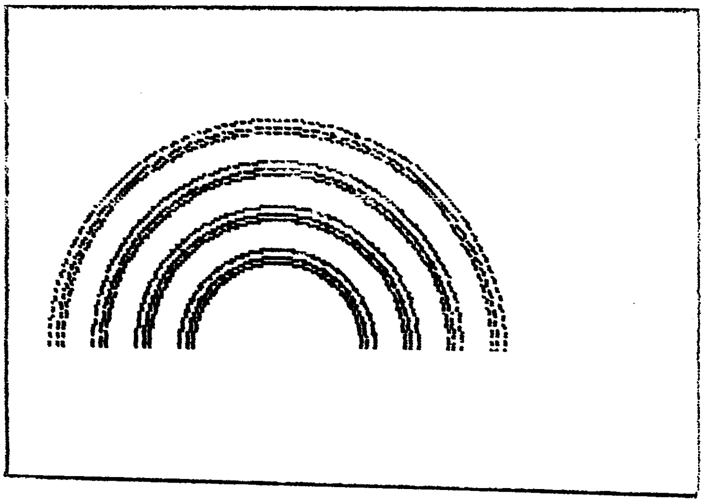
This program demonstrates how colours can be used with the high resolution plotting commands plus some of the limitations of high resolution colour. The display is a rainbow of four different coloured semicircles — red, yellow, green and blue. Each coloured semicircle is composed of three high resolution half circle plots. As the program stands the display produced has the four arcs each with a different colour, but notice that the gap between each arc is quite wide, try reducing the width of this gap and the colours of each arc start to break up. The gap can be reduced by changing the step value in line 200. The reason for this problem is simply that the colours are defined on a character square basis, trying to display two high resolution points of different colours in the same character space is impossible, the result is that the colour of the first plotted point will be changed to that of the second as soon as the second is plotted.

RUNNING THE PROGRAM

This program requires no input parameters, therefore simply enter RUN and watch the computer draw a coloured rainbow on the screen.

PROGRAM STRUCTURE

90	draw border around screen using subroutine at 500
110	coordinates of semicircle centre
120	start and end angle of semicircle
130	dot spacing in drawing semicircle
140 - 160	convert angles to radians
200	loop to draw four coloured arcs
210	get arc colour from data table
220	set to plot in that colour
240	loop to draw three lines in each arc
250 - 330	draw arc
410	colour data stored as colour values for each arc
500 - 560	border drawing subroutine



```
   1 REM RAINBOW
   2 REM ***********************
   3 REM
  10 REM program wil draw a colo
ured rainbow
  20 REM on the screen using hig
h resolution
  30 REM plotting.
  80 REM draw border around scre
en
  90 GO SUB 500
  95 REM
 100 REM set constants
 105 REM
 110 LET xo=100: LET yo=50
 120 LET p1=1: LET p2=180
 130 LET dp=1
 140 LET dp=dp*3.14159/180
 150 LET p1=p1*3.14159/180
 160 LET p2=p2*3.14159/180
 170 REM
 180 REM loop to draw four colou
r rainbow
 190 REM
 200 FOR r=30 TO 80 STEP 10
 210 READ c
 220 INK c
 225 REM
 230 REM three lines to each col
our
 235 REM
 240 FOR q=1 TO 3
 250 LET r=r+q
 260 FOR p=p1 TO p2 STEP dp
 270 LET x=r*COS (p)
 280 LET y=r*SIN (p)
 290 LET x=xo+x
 300 LET y=yo+y
 310 PLOT x,y
 320 NEXT p
 330 NEXT q
 340 NEXT r
 390 REM
 400 REM colour data for rainbow
 405 REM
 410 DATA 2,6,4,1,7
 500 REM border drawing subrouti
ne
 510 PLOT 0,0
 520 DRAW 255,0
 530 DRAW 0,175
 540 DRAW -255,0
 550 DRAW 0,-175
 560 RETURN
```

FAN

DESCRIPTION

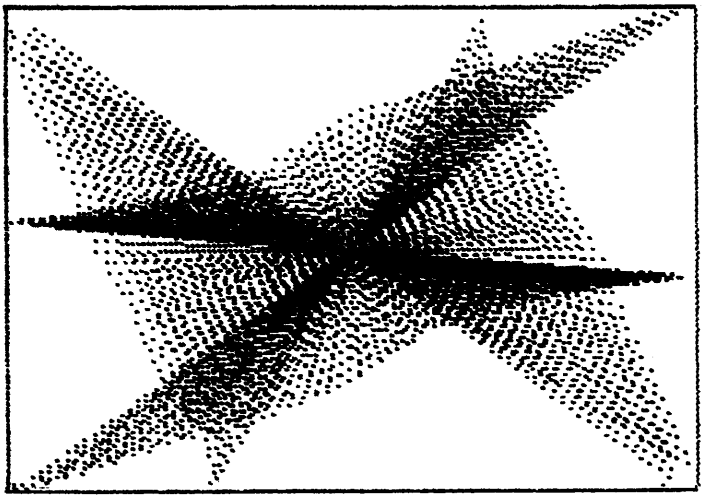
This is the last program in the section on colour and it simply produces a pretty changing and colourful pattern using high resolution colour plotting. The pattern is built up from different coloured high resolution lines and can be varied by changing the initial variable values in line 110 or by inserting extra loops into the main display loop — lines 140 to 210. The colour of each plotted line is set by a random value between 1 and 7 in lines 350-370. The lines are drawn by the subroutine 400 to 510.

RUNNING THE PROGRAM

Since no parameters are input by the program, simply type RUN and watch the pattern develop on the screen in constantly changing colours.

PROGRAM STRUCTURE

50	set background colour
110	initialisation variables — change for new pattern
120	initialise random seed
130	set starting ink colour for first line
140-220	main display loop — each of the four sub loops in this section draws a different part of the pattern, adds more sections or change values to change patterns
300-350	set values for line draw subroutine
360-370	set new line drawing colour
400-510	line drawing subroutine
600-660	border drawing subroutine



```
   1 REM FAN
   2 REM ***********************
   3 REM
  10 REM this program draws a
  20 REM coloured rotating fan
  30 REM
  40 REM set background colour
  50 PAPER 7
  60 REM
  70 REM draw border around scre
en
  80 GO SUB 600
  90 REM
 100 REM set up variables
 110 LET x=0: LET y=0: LET q=25:
 LET z=0
 120 RANDOMIZE
 130 INK 2
 133 REM
 135 REM main loop
 137 REM
 140 FOR x=5 TO 250 STEP 4
 150 GO SUB 300: NEXT x
 160 FOR y=5 TO 170 STEP 4
 170 GO SUB 300: NEXT y
 180 FOR x=250 TO 5 STEP -4
 190 GO SUB 300: NEXT x
 200 FOR y=170 TO 5 STEP -4
 210 GO SUB 300: NEXT y
 220 GO TO 140
 290 REM
 300 REM draw line and set ink c
olour
```

```
305 REM
310 LET xb=250-x: LET yb=170-y
320 LET xe=x: LET ye=y
330 GO SUB 400
340 LET z=z+1: IF z>=q THEN LET
z=0
350 LET q=INT (RND*50)
360 LET c=INT (RND*7)
370 INK c
380 RETURN
390 REM
400 REM draw line
405 REM
410 LET a=xe-xb
420 LET b=ye-yb
430 LET q=SQR (a*a+b*b)
440 LET ux=a/q
450 LET uy=b/q
460 FOR l=0 TO q STEP 4
470 LET x=xb+l*ux
480 LET y=yb+l*uy
490 PLOT x,y
500 NEXT l
510 RETURN
590 REM
600 REM draw border around scre
en
605 REM
610 PLOT 0,0
620 DRAW 255,0
630 DRAW 0,175
640 DRAW -255,0
650 DRAW 0,-175
660 RETURN
```

HIGH RESOLUTION GRAPHICS

HIGH RESOLUTION DISPLAYS

Any screen display by the computer, whether graphics or text, is built up from small dots known as pixels. The screen is 256 pixels wide and 176 pixels deep. The dots are formed as the electron beam which scans the television tube turns off and on, when it is on it excites the phosphor coating of the screen thereby generating a bright point of light. In the normal text mode each character is made up from an 8×8 square of dots.

A section of memory, the display file, is used to store the state of each dot on the screen. A '0' bit in memory signifies that the dot is displayed in the background colour and a '1' that it is in the foreground INK colour. In the text display mode the data for each character is obtained by the system software from a character data table in the operating system ROM. This is where the data for each of the displayable characters is stored, or if user definable characters have been created then the data is obtained from the RAM area allocated to storing these characters.

The display file, a section of the memory, is used to store the state of each dot on the screen. '0' bit in memory signifies that the dot is displayed in the background colour and '1' that it is in the of an 8×8 square of dots, and the data on these 64 dots can thus be stored in 8 bytes of memory, one byte for each row of dots. As the TV tube's electron beam scans from left to right repeatedly from top to bottom it builds up the 32 characters in each row one row of dots at a time. The data of each character is therefore not stored together in memory as eight bytes but at intervals 32 bytes apart. The display file can thus be divided into blocks of 256 bytes — this is the memory required to store one character line of display. The full screen display is further divided into four sections, the first block comprises lines 0 to 7, the second lines 8 to 15, the third lines 16 to 23 and the fourth contains just line 24. The reason for this is associated with scrolling and the protection of line 24 which is used for command displays.

Obviously one could display a point on the screen by using the POKE command to set the required bit in the display file, but the calculation of the required address makes this a slow and cumbersome operation. This method would be employed if a machine code was being used to generate the display, but in Basic it is far easier to use one of the Basic commands provided. There are four graphics commands in Basic:—

PLOT x, y — this plots a single pixel at the specified x and y coordinates.

DRAW x, y — a straight line is drawn from the last plotted point to a point specified in the coordinate parameters following the DRAW command. The problem with the DRAW command is that it is relative to the last plotted point rather than using absolute coordinates. The DRAW command thus obtains the absolute end coordinates of the line by using the last plotted point as the beginning coordinates and adding to these the relative offset coordinates following the DRAW command. A further complication is that in order to determine whether the end of the line is closer to the origin than the last plotted point then the relative coordinates can be negative. For these reasons most of the programmes in this book use a more flexible line drawing subroutine.

CIRCLE x, y, r — this command draws a circle of radius r, and with centre coordinates x, y.

POINT (x, y) — the POINT function will return a value of 1 if the pixel at the specified coordinates is the ink colour and a 0 if it is the paper colour.

Any of the above commands can be used in conjunction with the INVERSE command to erase a dot, line or circle. The INVERSE command simply sets the pixel to the paper colour thereby erasing it. The command to erase a pixel is thus:

PLOT INVERSE 1; x, y

One point to note is that to erase a line drawn using the DRAW command, the line must be erased by the DRAW INVERSE command in the same direction as the line was originally drawn.

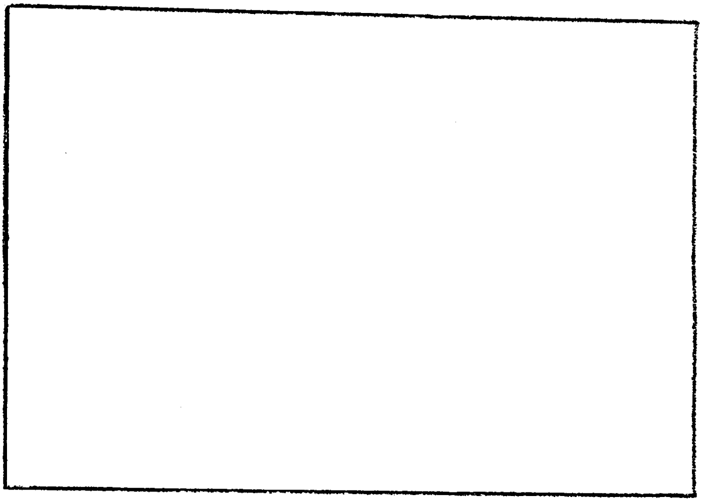
BORDER

DESCRIPTION

This simple program is probably one of the most frequently used programs in this book. This is because putting a thin border around the screen display area helps to neaten the display and draw the eye's attention to the text or display within the border. The routine is very short and simply draws four lines, two horizontal and two vertical. Starting at the bottom left hand corner of the screen, each line having as its starting point the end of the previous line.

RUNNING THE PROGRAM

This program requires no input parameters, just type RUN and it will draw a border around the screen.



```
  1 REM BORDER
  2 REM ***********************
  3 REM
 10 REM routine to draw a
 20 REM border around the
 30 REM edge of the screen
100 INK 0
110 PAPER 7
200 PLOT 0,0
210 DRAW 255,0
220 DRAW 0,175
230 DRAW -255,0
240 DRAW 0,-175
```

RECTANGLE 1

DESCRIPTION

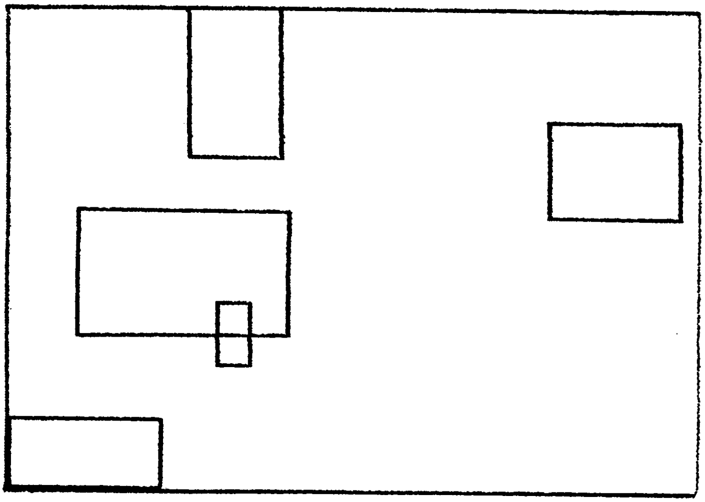
This routine is a natural extension of the program BORDER. Instead of just drawing a border around the screen, this routine will draw a border or box of any size and at any location on the screen. The program is in two sections, the first inputs the starting position of the bottom left hand corner of the box plus its height and width. These input variables are then checked to ensure that they are within the limits of the screen display area, ie: that the user is not trying to draw a box which extends over the edge of the screen. If the input variables are outside the limits then they are set to either the maximum or minimum default values. The second part of the program starts at line 270 and draws the box. To make the display neater a border is drawn around the screen display area by the subroutine at line 400.

RUNNING THE PROGRAM

This program requires the input of two pairs of parameters, the first pair are the X and Y coordinates of the bottom left corner of the rectangle. The second pair of values are the height and width of the rectangle.

PROGRAM STRUCTURE

40-50	set colours
120	draw border around screen using subroutine at 500
160	input X and Y coordinates of bottom left corner
190	input height and width of rectangle
230-300	check that the rectangle lies within the limits of screen
400-470	draw rectangle
500-560	border drawing subroutine



```
   1 REM RECTANGLE 1
   2 REM ***********************
   3 REM
  10 REM variable size rectangle
  20 REM drawing routine.
  30 REM set colours
  40 INK 0
  50 PAPER 7
 100 REM draw border around
 110 REM screen.
 120 GO SUB 500
 130 REM input bottom left
 140 REM corner coordinates of
 150 REM rectangle.
 160 INPUT x,y
 170 REM input rectangle height
 180 REM and width.
 190 INPUT h,w
 200 REM check input variables
 210 REM are within limits of
 220 REM maximum screen size.
 230 IF x>254 THEN LET x=254
 240 IF x<1 THEN LET x=1
 250 IF y>174 THEN LET y=174
 260 IF y<1 THEN LET y=1
 270 IF w+x>255 THEN LET w=255-x
 280 IF h+y>175 THEN LET h=175-y
 290 IF w<0 THEN LET w=0
 300 IF h<0 THEN LET h=0
 400 REM routine to draw
 410 REM rectangle.
 420 PLOT x,y
 430 DRAW w,0
 440 DRAW 0,h
 450 DRAW -w,0
 460 DRAW 0,-h
 470 GO TO 160
 500 REM border drawing routine
 510 PLOT 0,0
 520 DRAW 255,0
 530 DRAW 0,175
 540 DRAW -255,0
 550 DRAW 0,-175
 560 RETURN
```

RECTANGLE 2

DESCRIPTION

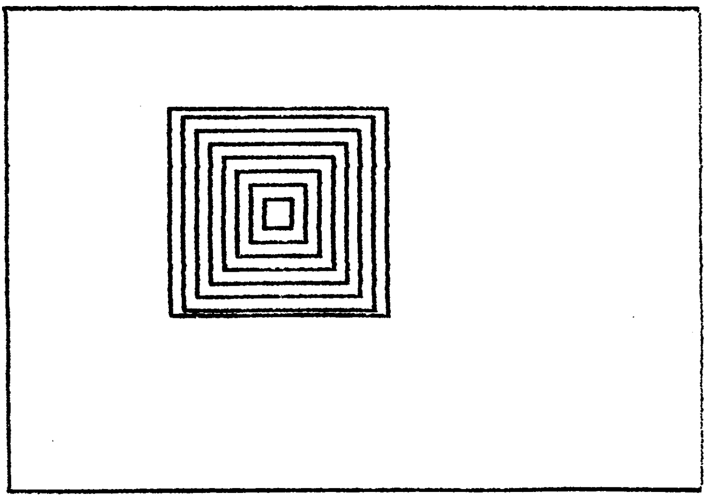
This is a variation of the program RECTANGLE 1, the difference is that it is the coordinates of the centre of the box which are input rather than the bottom left coordinates. The two programs are virtually identical except for the addition of lines 270 to 310 which simply convert the centre coordinates to the bottom left co-ordinates. Using the coordinates of the centre of a displayed shape is the conventional method of positioning a shape on the screen.

RUNNING THE PROGRAM

This program requires the input of two pairs of parameters, the first pair are the X and Y coordinates of the centre of the rectangle. The second pair of values are the height and width of the rectangle.

PROGRAM STRUCTURE

40-50	set colours
120	draw border around screen using subroutine at 500
160	input X and Y coordinates of rectangle centre
190	input height and width of rectangle
230-300	check that the rectangle lies within the limits of screen
340-370	convert centre coordinates to bottom left corner co-ordinates
400-470	draw rectangle
500-560	border drawing subroutine



```
  1 REM RECTANGLE2
  2 REM ***********************
  3 REM
 10 REM variable size rectangle
 20 REM drawing routine.
 30 REM set colours
 40 INK 0
 50 PAPER 7
100 REM draw border around
110 REM screen.
120 GO SUB 500
130 REM input centre
140 REM coordinates of
150 REM rectangle.
160 INPUT x,y
170 REM input rectangle height
180 REM and width.
190 INPUT h,w
200 REM check input variables
210 REM are within limits of
220 REM maximum screen size.
230 IF x>254 THEN LET x=254
240 IF x<1 THEN LET x=1
250 IF y>174 THEN LET y=174
260 IF y<1 THEN LET y=1
270 IF w+x>255 THEN LET w=255-x
280 IF h+y>175 THEN LET h=175-y
290 IF w<0 THEN LET w=0
300 IF h<0 THEN LET h=0
310 REM convert centre
320 REM cordinates to bottom
330 REM left corner coordinates
340 LET p=w/2
350 LET q=h/2
360 LET x=x-p
370 LET y=y-q
400 REM routine to draw
410 REM rectangle.
420 PLOT x,y
430 DRAW w,0
440 DRAW 0,h
450 DRAW -w,0
460 DRAW 0,-h
470 GO TO 160
500 REM border drawing routine
510 PLOT 0,0
520 DRAW 255,0
530 DRAW 0,175
540 DRAW -255,0
550 DRAW 0,-175
560 RETURN
```

RECTANGLE 3

DESCRIPTION

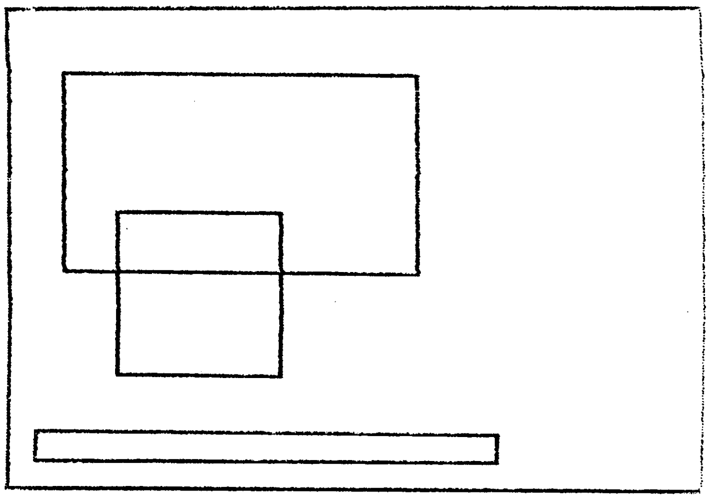
A rectangle can also be generated from two sets of coordinate points, these points being the two diagonally opposite corners of the rectangle. It should be noted, however, that like the previous two rectangle drawing routines this program will only draw a rectangle with sides parallel to the X and Y axis.

RUNNING THE PROGRAM

This program requires the input of two pairs of parameters, the first pair are the X and Y coordinates of the bottom left corner. The second pair of values are the X and Y coordinates of the top right corner of the rectangle.

PROGRAM STRUCTURE

40-50	set colours
120	draw border around screen using subroutine at 500
160	input X and Y coordinates of bottom left corner
190	input X and Y coordinates of top right corner
230-300	check that the rectangle lies within the limits of screen
400-470	draw rectangle
500-560	border drawing subroutine



```
  1 REM RECTANGLE 3
  2 REM ***********************
  3 REM
 10 REM variable size rectangle
 20 REM drawing routine.
 30 REM set colours
 40 INK 0
 50 PAPER 7
100 REM draw border around
110 REM screen.
120 GO SUB 500
130 REM input bottom left
140 REM corner coordinates of
150 REM rectangle.
160 INPUT x,y
170 REM input top right corner
180 REM coordinates.
190 INPUT x2,y2
200 REM check input variables
210 REM are within limits of
220 REM maximum screen size.
230 IF x>254 THEN LET x=254
240 IF x<1 THEN LET x=1
250 IF y>174 THEN LET y=174
260 IF y<1 THEN LET y=1
270 IF x2>255 THEN LET x2=255
280 IF y2>175 THEN LET y2=175
290 IF x2<1 THEN LET x2=1
300 IF y2<1 THEN LET y2=1
400 REM routine to draw
410 REM rectangle.
420 PLOT x,y
430 DRAW (x2-x),0
440 DRAW 0,(y2-y)
450 DRAW -(x2-x),0
460 DRAW 0,-(y2-y)
470 GO TO 160
500 REM border drawing routine
510 PLOT 0,0
520 DRAW 255,0
530 DRAW 0,175
540 DRAW -255,0
550 DRAW 0,-175
560 RETURN
```

BARCHART

DESCRIPTION

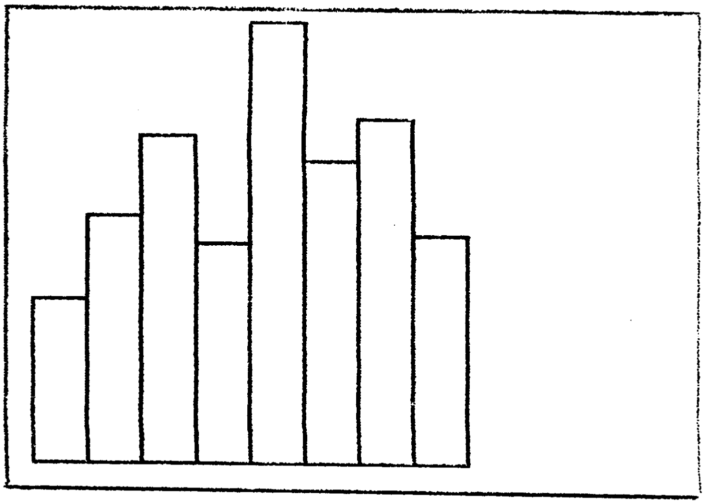
This program shows one application for a routine to draw a variable sized rectangle. The program draws a barchart using the data in the data statements in lines 100-160. The program repeatedly draws boxes of variable height until the data statement containing zero is reached, whereupon the routine terminates. The X, Y and W values are set as constants at the beginning of the program. The Y and W values will stay constant but the X value is incremented by W for each bar displayed. By changing the constants the position, size and maximum number of bars displayed can be varied.

RUNNING THE PROGRAM

This program requires no input parameters, simply type RUN and it will display a barchart on the screen using the values in the data statements.

PROGRAM STRUCTURE

50	draw border around screen using subroutine at 500
80-90	set colours
110	position of start of first bar from left border edge
120	height above bottom border
130	border width
150	read bar height from data table
200-250	draw bar
270	calculate bottom left corner of next bar
340-420	data for bar heights, table terminated by 999
500-560	border drawing subroutine



```
   1 REM BARCHART
   2 REM **********************
   3 REM
  10 REM program to draw a
  20 REM barchart using the
  30 REM rectangle drawing routi
ne
  40 REM draw border around scre
en
  50 GO SUB 500
  70 REM set colours
  80 INK 0
  90 PAPER 7
 100 REM set constants
 110 LET x=10: REM start of firs
t bar from left border edge
 120 LET y=10: REM height above
bottom border
 130 LET w=20: REM bar width
 140 REM get bar height data
 150 READ h
 160 IF h=999 THEN STOP
 200 REM draw bar
 210 PLOT x,y
 220 DRAW w,0
 230 DRAW 0,h
 240 DRAW -w,0
 250 DRAW 0,-h
 260 REM next bar
 270 LET x=x+w
 280 GO TO 150
 300 REM data for height of each
 310 REM bar in barchart. value
 320 REM of 999 indicates end of
 330 REM data.
 340 DATA 60
 350 DATA 90
 360 DATA 120
 370 DATA 80
 380 DATA 160
 390 DATA 110
 400 DATA 126
 410 DATA 63
 420 DATA 999
 500 REM draw border around scre
en
 510 PLOT 0,0
 520 DRAW 255,0
 530 DRAW 0,175
 540 DRAW -255,0
 550 DRAW 0,-175
 560 RETURN
```

BLOCK

DESCRIPTION

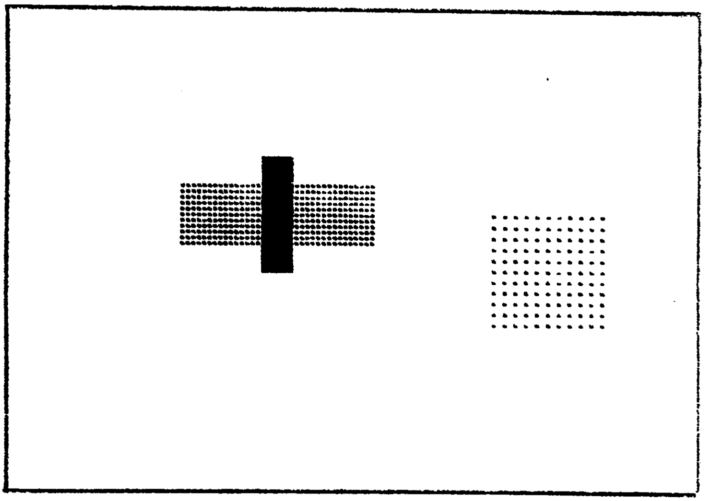
The POINT plot command is the most useful of the high resolution commands and can be used in a wide range of applications. A very simple application is shown in this programme, it fills a rectangular block of the screen with dots, the density of the dots being variable. This can be used in a range of applications from the next program — BARCHART, to simple shading of areas of the screen. By careful use of ink and paper colour the creation of new colours — red dots on a yellow background will at a distance give the appearance of an orange colour.

RUNNING THE PROGRAM

The program requires the input of five variables. The first two are input in line 110 they are the X and Y coordinates of the bottom right corner of the block. The next two are input in line 120 and they are the width of the block and the height of the block. The height and width are both parallel to the Y and X coordinate axis of the screen, it is impossible using this routine to draw a block at an angle to the screen axis. The last variable is input in line 130 and is the spacing between dots in the block, both vertical and horizontal. If the dot spacing is 1 then a solid block of dots is drawn, a dot spacing of 2 will put a space between each dot, and similarly for other dot spacing values. Note all these values are the number of pixels, either from the bottom left corner of the screen or over the specified distance (remember one character space is 8 pixels high and 8 pixels wide). To vary the block colours change the values in lines 60 and 70 of the program.

PROGRAM STRUCTURE

60-70 set colours for plotting
90 draw border around screen using subroutine at 300
110-130 input parameters for block size and position
210-290 block drawing routine
300-360 border drawing subroutine



```
   1 REM BLOCK
   2 REM ***********************
   3 REM
  10 REM program to draw a recta
ngular
  20 REM block of variable densi
ty
  30 REM dots on the screen
  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  80 REM draw border around scre
en
  90 GO SUB 300
  95 REM
 100 REM input block drawing par
ameters
 105 REM
 110 INPUT x,y: REM bottom left
corner coordinates
 120 INPUT w,h: REM width and he
ight
 130 INPUT ds: REM dot spacing
 195 REM
 200 REM draw block
 205 REM
 210 LET xb=x-w/2
 220 LET xe=x+w/2
 230 LET yb=y-h/2
 240 LET ye=y+h/2
 250 FOR x=xb TO xe STEP ds
 260 FOR y=yb TO ye STEP ds
 270 PLOT x,y
 280 NEXT y
 290 NEXT x
 295 GO TO 100
 300 REM draw border
 305 REM
 310 PLOT 0,0
 320 DRAW 255,0
 330 DRAW 0,175
 340 DRAW -255,0
 350 DRAW 0,-175
 360 RETURN
```

BARCHART

DESCRIPTION

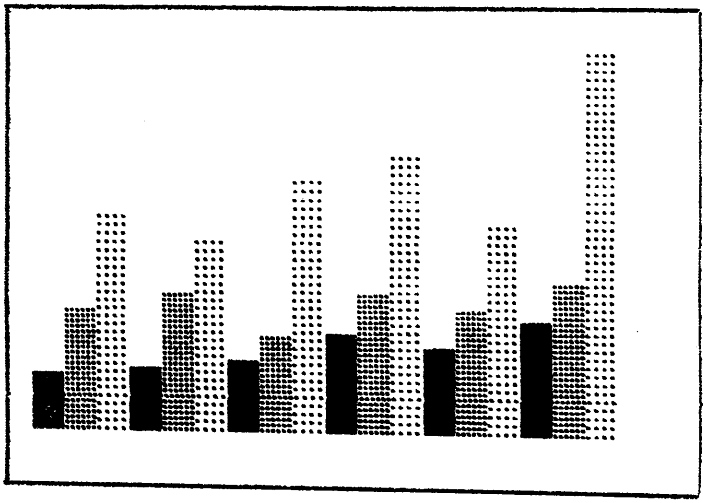
The barchart is a very useful way of displaying data and comparing two or more sets of related data, and blocks of variable density dots make an ideal way of displaying the bars and differentiating between different sets of data. The example barchart used in this program is derived from a data table (lines 130-155), but could just as easily have been input from the keyboard, derived from calculations or retrieved from a data storage device. There are three different sets of data in the example which are differentiated by having a dot spacing respectively of 1, 2 and 3.

RUNNING THE PROGRAM

There are no input variables required by this program since the data for generating the barchart is stored as data statements within the program.

PROGRAM STRUCTURE

60-70 set colours for plotting
90 draw border around screen using subroutine at 400
130-170 data for generating the barchart, it is stored as bar height in pixels followed by a dot density value. This determines the data set of the bar. In the example there are three sets of data each having six values. The end of the data table is signalled by putting 0 for each value.
210 the value of B sets the bottom left corner X coordinate of each bar. Change this value to move the chart across the screen.
220-300 block display routine, note that in line 250 the bottom Y axis (start of the chart) is set to 20 pixels up from the bottom of the screen; to move the chart up or down the screen change this value. Line 240 and 290 determine the width of each bar; the plotting is 10 pixels wide and the start of the next bar is 12 pixels from the start of the previous bar (this leaves a slight space between bars). To vary the bar width change these values.
400-460 border drawing subroutine.



```
   1 REM BARCHART
   2 REM ************************
   3 REM
  10 REM program to draw a barch
art
  20 REM using variable density
blocks
  30 REM to differentiate betwee
n diferent
  40 REM sets of data.
  45 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 400
  95 REM
 100 REM barchart data
 110 REM data stored as bar heig
ht
 120 REM followed by dot density
.
 130 DATA 20,1,45,2,80,3
 135 DATA 22,1,50,2,70,3
 140 DATA 25,1,35,2,90,3
 145 DATA 35,1,50,2,100,3
 150 DATA 30,1,45,2,75,3
 155 DATA 40,1,55,2,140,3
 160 REM data terminated by doub
le zero
 170 DATA 0,0
 195 REM
 200 REM draw barchart
 205 REM
 210 LET b=10: REM start positio
n from left screen margin.
 220 READ h,d
 230 IF d=0 THEN STOP
 240 FOR x=b TO b+10 STEP d
 250 FOR y=20 TO h+20 STEP d
 260 PLOT x,y
 270 NEXT y
 280 NEXT x
 290 LET b=b+12: REM set start o
f next bar
 300 GO TO 220
 400 REM draw border
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

LINE

DESCRIPTION

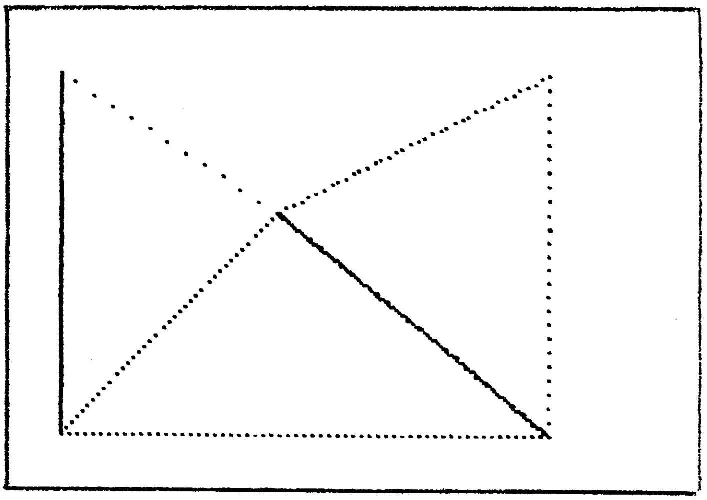
Although the computer command DRAW will draw a high resolution line between two points on the screen it has several serious drawbacks. Foremost of these drawbacks is that it uses relative coordinates, which are not very easy to use in many graphics applications. Another drawback is that it is impossible to draw a line with variable spacing between the dots. Both these problems are overcome by using this program, although it has one shortcoming in that since it is written in Basic it is rather slow. Most of the programs in this book which require line drawing use this routine. Two versions of this program are given, the first 'LINE' simply draws a line of specified dot separation between two sets of coordinates. The second is identical except that the variable R$ is input to determine if the line is to be drawn or erased (the line is erased if R$ = E).

RUNNING THE PROGRAM

In the program 'LINE' there are five variables which are input by program lines 100 to 130. The first two are input by line 100 and are the X, Y coordinates of the beginning of the line. The second two variables are the X and Y coordinates of the end of the line, and the last variable is the spacing between the dots used to draw the line. The program 'LINE2' has an extra variable input in line 120 this determines whether the line is drawn or erased, if an 'E' is input then the line will be erased, any other letter then the line will be drawn.

PROGRAM STRUCTURE

60-70 set colours
90 draw border around screen using subroutine at 400
100-130 input variables for start and end of line coordinates and dot spacing
150-240 line drawing routine
400-460 border drawing subroutine



```
   1 REM LINE
   2 REM ************************
   3 REM
  10 REM this program draws a li
ne
  20 REM between two sets of coo
rdinates
  30 REM the spacing between the
 dots used
  40 REM is variable.
  45 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM line drawing routine pa
rameter input
 110 INPUT xb,yb: REM cordinates
 of beginning of line
 120 INPUT xe,ye: REM cordinates
 of endof line
 130 INPUT ds: REM dot spacing
 135 REM
 140 REM draw line
 145 REM
 150 LET p=xe-xb
 160 LET q=ye-yb
 170 LET r=SQR (p*p+q*q)
 180 LET lx=p/r
 190 LET ly=q/r
 200 FOR i=0 TO r STEP ds
 210 LET x=xb+i*lx
 220 LET y=yb+i*ly
 230 PLOT x,y
 240 NEXT i
 300 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

```
   1 REM LINE 2
   2 REM ***********************
   3 REM
  10 REM this program draws or e
rases a line
  20 REM between two sets of coo
rdinates
  30 REM the spacing between the
 dots used
  40 REM is variable.
  45 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM line drawing routine pa
rameter input
 105 INPUT xb,yb: REM coordinate
s of beginning of line
 110 INPUT xe,ye: REM coordinate
s of end of line
 115 INPUT ds: REM dot spacing
 120 INPUT r$: REM draw or erase
 125 REM
 130 REM draw line
 135 REM
 140 LET p=xe-xb
 145 LET q=ye-yb
 150 LET r=SQR (p*p+q*q)
 155 LET lx=p/r
 160 LET ly=q/r
 165 FOR i=0 TO r STEP ds
 170 LET x=xb+i*lx
 175 LET y=yb+i*ly
 180 IF r$="e" THEN GO TO 195
 185 PLOT x,y
 190 GO TO 200
 195 PLOT INVERSE 1;x,y
 200 NEXT i
 300 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

RECTANGLE 4

DESCRIPTION

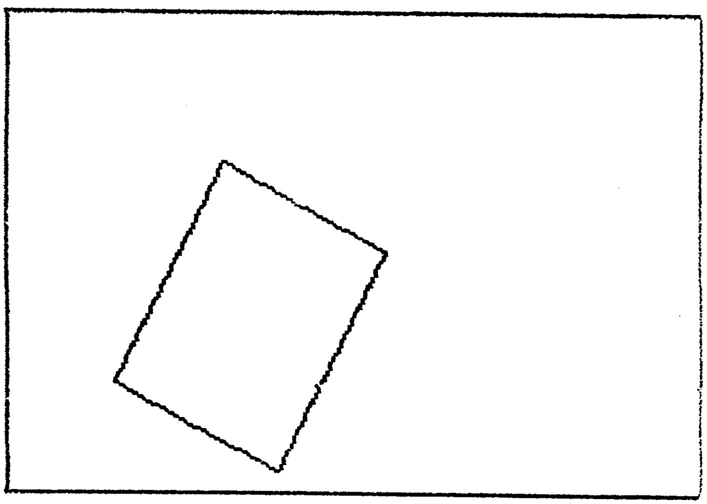
Whereas the programs 'RECTANGLE 1 to RECTANGLE 3' are only able to draw rectangles with sides parallel to the X and Y axis of the screen this program shows how to draw rectangles with sides which are not parallel to the screen axis. This is simply done by using a matrix of coordinates. Matrices are very important in graphics and an understanding of the principles is essential. The coordinate matrix is usually stored as data statements within the program and subsequently placed in an array. The values in this array can be manipulated mathematically, thereby allowing the shape to be rotated, scaled, or moved about the screen area. All these will be dealt with in later sections of this book. In this program the values are simply used to display the shape at the specified coordinates.

RUNNING THE PROGRAM

Since all the coordinate values are stored in data statements – lines 210 and 220, there are no values to be input in the program. However, to change the size or position of the rectangle it is necessary to input new data values into these data statements. Five coordinate values are required to draw the four lines of the rectangle, the X component of these five coordinates is stored in line 210 and the corresponding Y component in line 220. The best way to obtain these coordinate values for a new rectangle is to draw the shape with the correct scale and orientation onto graph paper and measure the required values.

PROGRAM STRUCTURE

50-60	set colours
80	draw border around screen using subroutine at 600
110-180	load matrix data into arrays
210	data for X component of coordinates
220	data for Y component of coordinates
300-350	set variables for line draw subroutine
400-510	draw line. Note: dot spacing in 410 is set to 1
600-660	border drawing subroutine



```
   1 REM RECTANGLE 4
   2 REM   *********************
   3 REM
  10 REM program to draw a recta
ngle
  20 REM using matrix methods.
  30 REM
  40 REM set colours.
  50 INK 0
  60 PAPER 7
  70 REM draw border around scre
n
  80 GO SUB 600
 100 REM input data from data st
atements
 110 REM into array.
 120 DIM m(5,2)
 130 FOR c=1 TO 5
 140 READ m(c,1)
 150 NEXT c
 160 FOR c=1 TO 5
 170 READ m(c,2)
 180 NEXT c
 200 REM data for coordinates
 205 REM
 210 DATA 40,80,140,100,40
 220 DATA 40,120,88,8,40
 300 REM draw rectangle
 310 FOR c=1 TO 4
 320 LET xb=m(c,1)
 330 LET yb=m(c,2)
 340 LET xe=m(c+1,1)
 350 LET ye=m(c+1,2)
 360 GO SUB 400
 370 NEXT c
 380 STOP
 400 REM line drawing routine
 405 REM
 410 LET ds=1: REM dot spacing
 420 LET p=xe-xb
 430 LET q=ye-yb
 440 LET r=SQR (p*p+q*q)
 450 LET lx=p/r
 460 LET ly=q/r
 470 FOR i=0 TO r STEP ds
 480 LET x=xb+i*lx
 490 LET y=yb+i*ly
 500 PLOT x,y
 510 NEXT i
 600 REM border drawing routine
 610 PLOT 0,0
 620 DRAW 255,0
 630 DRAW 0,175
 640 DRAW -255,0
 650 DRAW 0,-175
 660 RETURN
```

POLYGON

DESCRIPTION

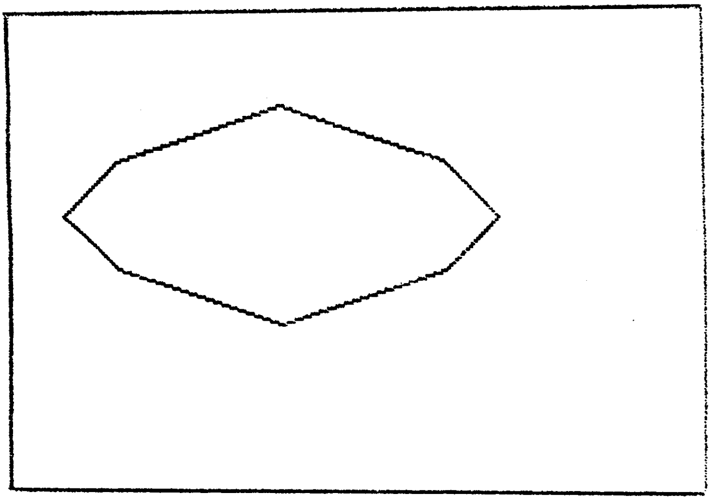
The only difference between this program and the previous program 'RECTANGLE 4' is the data used to draw the shape. The reason being that the use of a coordinate matrix is not confined to rectangles, it can be used to generate any required shape. In this program the data will draw an irregularly shaped octagon. To change the shape and its position simply change the data.

RUNNING THE PROGRAM

The coordinate data values are stored as data statements — lines 210 and 220, so now there are no values to be input when the program is run. The size, shape or position on the screen of the shape can be changed by changing the data values in the data statements. It should be noted that when a shape is drawn the number of pairs of coordinate values is one more than the number of lines in the shape. The number of coordinate values used to draw the shape is stored as the first data statement value — line 205. The coordinates are stored as two sets of data, first all the X values and then in corresponding order all the Y values. In the example the X coordinates are thus stored in the data statement on line 210 and the Y values in line 220.

PROGRAM STRUCTURE

50-60	set colours
80	draw border around screen using subroutine at 600
110-180	load matrix data into arrays
205	number of coordinates in matrix data
210	data for X component of coordinates
220	data for Y component of coordinates
300-350	set variables for line draw subroutine
400-510	subroutine to draw line
600-660	border drawing subroutine



```
   1 REM POLYGON1
   2 REM ***********************
   3 REM
  10 REM program to draw a polyg
on
  15 REM with N sides using matr
ix
  20 REM methods.
  40 REM set colours.
  50 INK 0
  60 PAPER 7
  70 REM draw border around scre
en
  80 GO SUB 600
 100 REM input data from data st
atements
 110 REM into array.
 115 READ n: REM number of sides
 120 DIM m(n,2)
 130 FOR c=1 TO n
 140 READ m(c,1)
 150 NEXT c
 160 FOR c=1 TO n
 170 READ m(c,2)
 180 NEXT c
 200 REM data for coordinates
 205 DATA 9
 210 DATA 20,40,100,160,180,160,
100,40,20
 220 DATA 100,80,60,80,100,120,1
40,120,100
 300 REM draw polygon
 310 FOR c=1 TO n-1
 320 LET xb=m(c,1)
 330 LET yb=m(c,2)
 340 LET xe=m(c+1,1)
 350 LET ye=m(c+1,2)
 360 GO SUB 400
 370 NEXT c
 380 STOP
 400 REM line drawing routine
 405 REM
 410 LET ds=1: REM dot spacing
 420 LET p=xe-xb
 430 LET q=ye-yb
 440 LET r=SQR (p*p+q*q)
 450 LET lx=p/r
 460 LET ly=q/r
 470 FOR i=0 TO r STEP ds
 480 LET x=xb+i*lx
 490 LET y=yb+i*ly
 500 PLOT x,y
 510 NEXT i
 600 REM border drawing routine
 610 PLOT 0,0
 620 DRAW 255,0
 630 DRAW 0,175
 640 DRAW -255,0
 650 DRAW 0,-175
 660 RETURN
```

POLYGON 2

DESCRIPTION

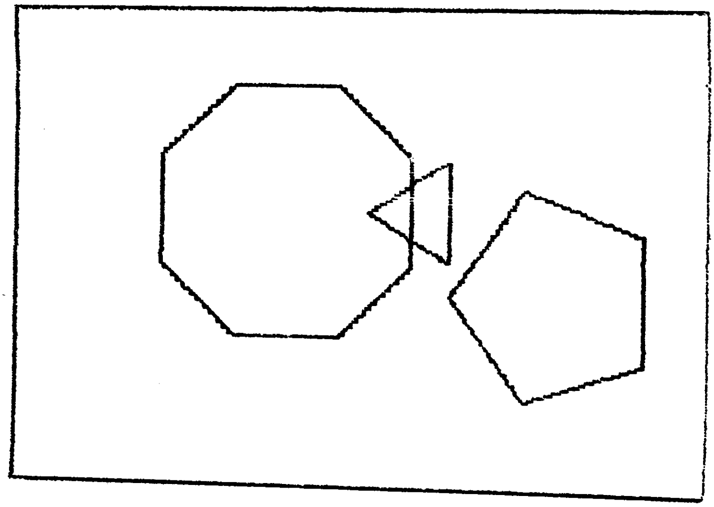
To save having to work out the end of line coordinates for each line of a polygon it is far easier, given a regular N sided polygon, to calculate these values within the program. This is done by the program POLYGON 2 which simply requires the centre of the polygon, the radius, the angular offset and the number of sides to the polygon. The program is configured to draw a series of polygons using data from a data table. The five parameters required to draw each polygon are then used to calculate a table of coordinates for each of the lines in the polygon, these values are then stored in the array m(n,2).

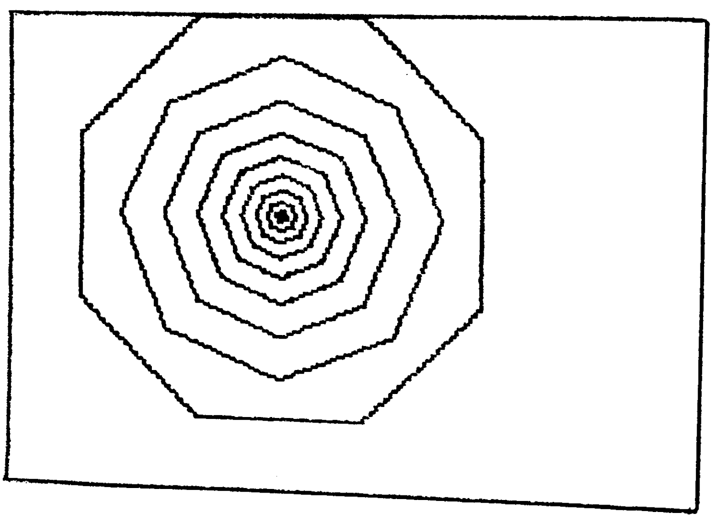
RUNNING THE PROGRAM

All the parameters required by the program are stored directly within the program. The X and Y coordinates of the central axis around which the shape is rotated is stored as the variables cx and cy. The number of lines in the shape is stored as variable n, r is the radius of the polygon and os is the angular offset. These values are stored as data statements in lines 300 to 320 (each line of data-statement holds the data for one polygon). To change the polygon's shape, orientation or position then change the values in the data statements, to add extra polygons then add further lines of data statement values.

PROGRAM STRUCTURE

60-70	set colours
90	draw border around screen using subroutine at 900
140	get data from data statement for next polygon
160-170	matrix for line coordinates and angles
180-190	convert angles to radians
200-220	calculate angles for each corner and put in array
300-320	data for drawing three polygons
400-460	calculate line coordinates and put in array
480-550	draw polygon
610-720	line drawing subroutine
800-860	border drawing subroutine





```
   1 REM POLYGON 2
   2 REM **********************
   3 REM
  10 REM program to draw N sided
 polygons
  20 REM given the centre, radiu
s,
  30 REM and angular offset.
  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  80 REM draw border
  90 GO SUB 800
 100 REM routine to draw polygon
 110 REM input parameters from d
ata
 120 REM statements and set up m
atrix array
 130 REM
 140 READ cx,cy,r,n,os
 150 LET q=n+1
 160 DIM m(q,2)
 170 DIM a(q)
 180 LET ad=2*3.14159/n
 190 LET os=os/180*3.14159
 200 FOR c=1 TO n
 210 LET a(c)=c*ad+os
 220 NEXT c
 300 DATA 100,100,50,8,22.5
 310 DATA 150,100,20,3,60
 320 DATA 200,70,40,5,36
 395 REM
 400 REM set coordinates
 405 REM
 410 FOR x=1 TO n
 420 LET m(x,1)=cx+r*COS (a(x))
 430 LET m(x,2)=cy-r*SIN (a(x))
 440 NEXT x
 450 LET m(n+1,1)=m(1,1)
 460 LET m(n+1,2)=m(1,2)
 465 REM
 470 REM draw polygon
 475 REM
 480 FOR c=1 TO n
 490 LET xb=m(c,1)
 500 LET yb=m(c,2)
 510 LET xe=m(c+1,1)
 520 LET ye=m(c+1,2)
 530 GO SUB 600
 540 NEXT c
 550 GO TO 100
 595 REM
 600 REM line drawing routine
 605 REM
 610 LET ds=1: REM dot spacing
 620 LET p=xe-xb
 630 LET q=ye-yb
 640 LET r=SQR (p*p+q*q)
```

```
650 LET lx=p/r
660 LET ly=q/r
670 FOR i=0 TO r STEP ds
680 LET x=xb+i*lx
690 LET y=yb+i*ly
700 PLOT x,y
710 NEXT i
720 RETURN
795 REM
800 REM border plot routine
805 REM
810 PLOT 0,0
820 DRAW 255,0
830 DRAW 0,175
840 DRAW -255,0
850 DRAW 0,-175
860 RETURN
```

RECTANGLE 5

DESCRIPTION

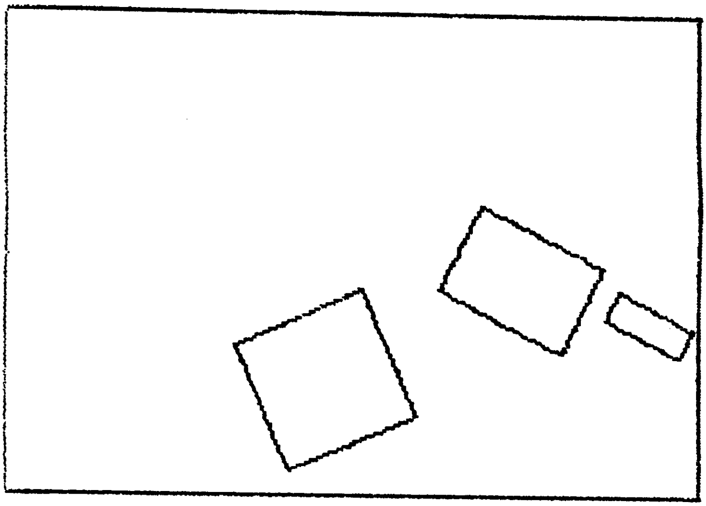
The problem with the programs RECTANGLE 1 to 3 is that they are unable to draw a rectangle with sides which are not parallel to the sides of the screen. Program RECTANGLE 4 overcame this but required the coordinates of all four corners. This program will draw rectangles of any orientation, given the coordinates of two corners and the length of one side, this is done using a simple calculation based on Pythagoras Theorem to calculate a matrix of corner coordinates.

RUNNING THE PROGRAM

The program requires the input of five parameter values. The first two are the X and Y coordinates of the bottom left corner and next two values are the coordinates of the bottom right corner. The last value is the length of a side at right angles to the side described by the pair of coordinates points.

PROGRAM STRUCTURE

50 - 60	set colours
80	draw border around screen using subroutine at 600
105	set up coordinate matrix array
110	input bottom left X,Y coordinates
120	input bottom right X,Y coordinates
130	input length of perpendicular side
140 - 295	calculate all corner coordinates of the rectangle
300 - 360	draw rectangle
400 - 510	line drawing subroutine
600 - 660	border drawing subroutine



```
   1 REM RECTANGLES
   2 REM *************************
   3 REM
  10 REM program to draw a recta
ngle
  15 REM given coordinates of tw
o corners
  20 REM and length of one side.
  30 REM
  40 REM set colours.
  50 INK 0
  60 PAPER 7
  70 REM draw border around scre
n
  80 GO SUB 600
 100 REM input data
 105 DIM m(5,2)
 110 INPUT x1,y1
 120 INPUT x2,y2
 130 INPUT l
 140 LET p=x2-x1
 150 LET q=y2-y1
 160 LET r=SQR (p*p+q*q)
 170 LET lx=p/r
 180 LET ly=q/r
 190 LET wx=-ly
 200 LET wy=lx
 210 LET m(1,1)=x1
 220 LET m(2,1)=x2
 230 LET m(3,1)=x2+wx*l
 240 LET m(4,1)=x1+wx*l
 250 LET m(5,1)=x1
 260 LET m(1,2)=y1
 270 LET m(2,2)=y2
 280 LET m(3,2)=y2+wy*l
 290 LET m(4,2)=y1+wy*l
 295 LET m(5,2)=y1
 300 REM draw polygon
 305 REM
 310 FOR c=1 TO 4
 320 LET xb=m(c,1)
 330 LET yb=m(c,2)
 340 LET xe=m(c+1,1)
 350 LET ye=m(c+1,2)
 360 GO SUB 400
 370 NEXT c
 380 GO TO 100
 400 REM line drawing routine
 405 REM
 410 LET ds=1: REM dot spacing
 420 LET p=xe-xb
 430 LET q=ye-yb
 440 LET r=SQR (p*p+q*q)
 450 LET lx=p/r
 460 LET ly=q/r
 470 FOR i=0 TO r STEP ds
 480 LET x=xb+i*lx
 490 LET y=yb+i*ly
 500 PLOT x,y
```

```
510 NEXT i
600 REM border drawing routine
610 PLOT 0,0
620 DRAW 255,0
630 DRAW 0,175
640 DRAW -255,0
650 DRAW 0,-175
660 RETURN
```

CIRCLE

DESCRIPTION

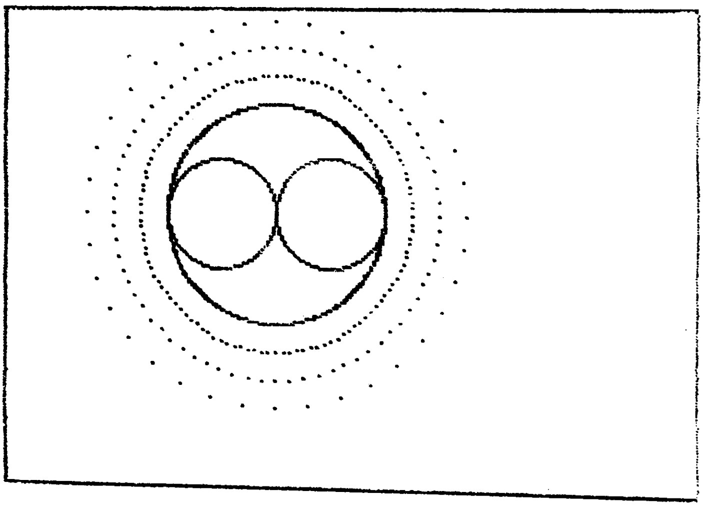
Plotting an ordinary circle with your computer is remarkably easy, using the built-in CIRCLE command, which allows you to specify the central X and Y coordinates, and also the radius. This will then plot a complete circle on the screen. However, for many applications we will not want a full circle, although we will require full image of the circle to be displayed. In other words, we want to be able to specify a distance between the points plotted that make up the circumference of the circle. The program CIRCLE does just that, by use of the PLOT command to plot each individual dot of the circumference to a specified separation. This is the variable DS in the program listing, line 130.

RUNNING THE PROGRAM

A number of inputs are required to get the program going. In line 110 we input the X and Y coordinates of the centre of the circle, namely XC and YC, followed in line 120 by the radius RA. Our fourth input is the separation between the dots as mentioned earlier, that is the variable DS in line 130. This dot separation is then converted in line 210 (by multiplying by PI and dividing by 180) to form the STEP for the FOR NEXT loop in line 230 which initiates the plotting process. As we know, 2 PI radians equal 360 degrees, and hence the statement in line 230. Then we just calculate the distance of the dot in terms of X and Y coordinates from the centre of the circle, and PLOT the point. Line 300 then sends us back for another run and another circle.

PROGRAM STRUCTURE

60-70	set colours
90	draw border around screen using subroutine at 400
110	input coordinates of circle centre
120	input circle radius
130	input dot separation
210-290	draw circle
300	back for another go
400-460	border drawing routine



```
   1 REM CIRCLE
   2 REM ************************
   3 REM
  10 REM routine to draw a circl
e
  20 REM spacing between the dot
s used
  30 REM to draw the circle is v
ariable
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM input circle drawing pa
rameters
 110 INPUT xc,yc: REM coordinate
s of circle centre
 120 INPUT ra: REM circle radius
 130 INPUT ds: REM dot spacing
 195 REM
 200 REM draw circle
 205 REM
 210 LET ds=ds*3.14159/180
 220 LET r=ra
 230 FOR p=0 TO 2*3.14159 STEP d
s
 240 LET x=r*COS (p)
 250 LET y=r*SIN (p)
 260 LET x=x+xc
 270 LET y=y+yc
 280 PLOT x,y
 290 NEXT p
 300 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

ELLIPSE

DESCRIPTION

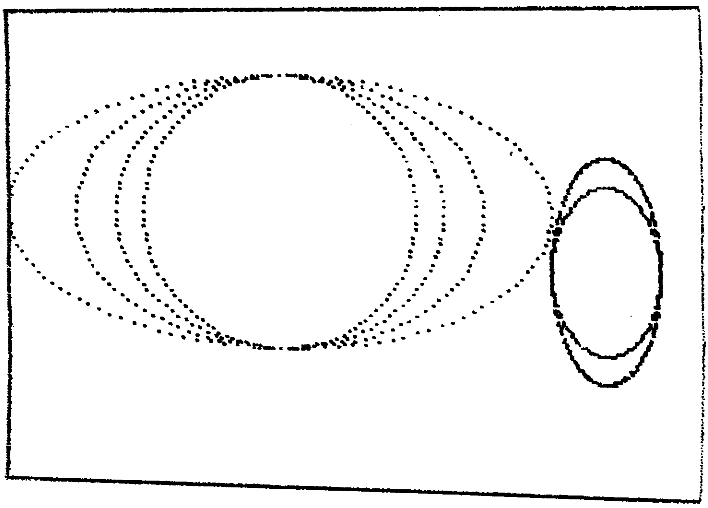
Your computer is equipped with the CIRCLE command to facilitate the drawing of a circle, but what it does not possess is a command to plot an ellipse. That is, a circle that is offset on two sides from the central point in either the X or the Y direction. Using the routine we developed in the program Circle, together with a couple of additions to handle the elliptical effect, we can plot an ellipse, or indeed any number of ellipses, with variable dot spacing. The offsets are specified in line 140, and determine the degree of ellipse. The variables OX and OY are used, and obviously if OX is zero we get an ellipse in the Y direction, and vice versa. Naturally we can give values to both of these to get a number of interesting effects.

RUNNING THE PROGRAM

In structure this is very similar to the Circle program earlier, but a couple of major differences are worthy of note. In line 140 we are asked to input the variables OX and OY to specify the degree of ellipse. These are subsequently used in our ellipse drawing routine in lines 240-250 to calculate precisely where our point is to be plotted. The rest of the program, including the routine to specify the separation of the dots (lines 210 and 230) is virtually the same.

PROGRAM STRUCTURE

60-70	set colours
90	draw border round screen using subroutine at 400
110	input coordinates of ellipse centre
120	input ellipse radius
130	input dot separation
140	input elliptical offsets in X and Y direction
210-290	draw ellipse
300	back for another go
400-460	border drawing routine



```
   1 REM ELLIPSE
   2 REM ************************
   3 REM
  10 REM routine to draw an elli
pse using offsets
  20 REM spacing between the dot
s used
  30 REM to draw the ellipse is
variable
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM input ellipse drawing p
arameters
 110 INPUT xc,yc: REM coordinate
s of ellipse centre
 120 INPUT ra: REM ellipse radiu
s
 130 INPUT ds: REM dot spacing
 140 INPUT ox,oy: REM elliptical
 offsets in x and y axis
 195 REM
 200 REM draw ellipse
 205 REM
 210 LET ds=ds*3.14159/180
 220 LET r=ra
 230 FOR p=0 TO 2*3.14159 STEP d
s
 235 REM
 240 LET x=r*COS (p)*ox
 250 LET y=r*SIN (p)*oy
 260 LET x=x+xc
 270 LET y=y+yc
 280 PLOT x,y
 290 NEXT p
 300 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

ARC1

DESCRIPTION

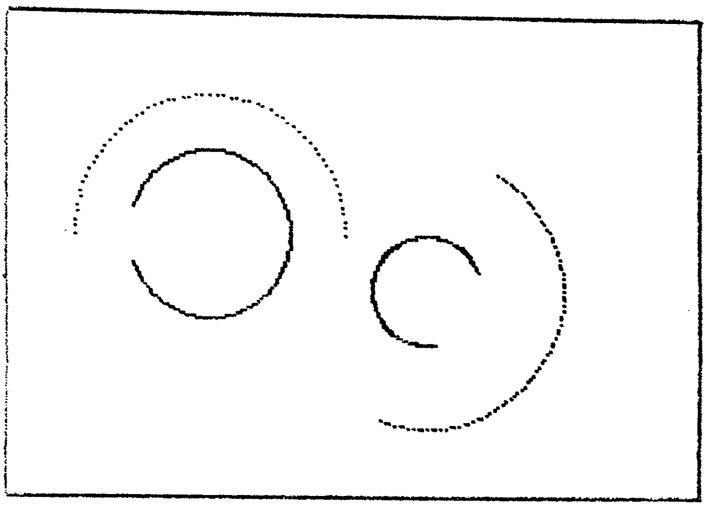
The computer command DRAW, whilst not being without its uses, suffers from a number of limitations. Like the CIRCLE command, you can only draw complete, filled in lines. Also, whether we use it in conjunction with the third parameter (other than X and Y co-ordinates of the finishing point), namely the angle through which it must turn, or not, we must always remember that DRAW will start off from the last point plotted by CIRCLE, PLOT or the previous DRAW statement. In order to draw an arc from anywhere to anywhere, and to be able to have user-definable dot spacing, the routines in the program ARC1 were developed.

RUNNING THE PROGRAM

A number of inputs are required. In line 110 we must specify XC and YC, that is, the centre of the arc. Line 120 allows us to specify RA, the arc radius, and line 130 lets us input the dot separation DS. Two further inputs in line 140 contain the crux of the matter, and give us that much needed flexibility over DRAW, by allowing us to specify the start and end angles of the arc. Thus, we are not limited in where we can start drawing. The drawing routine in lines 250 to 310 is similar to the ones in earlier programs in this series.

PROGRAM STRUCTURE

60-70	set colours
90	draw border round screen using subroutine at 400
110	input coordinates of arc centre
120	input arc radius
130	input dot separation
140	input start and end angles for arc
210-290	draw arc
320	back for another go
400-460	border drawing routine



```
   1 REM ARC 1
   2 REM ***********************
   3 REM
  10 REM routine to draw an arc
  20 REM spacing between the dot
s used
  30 REM to draw the arc is vari
able
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM input arc drawing param
eters
 110 INPUT xc,yc: REM coordinate
s of centre of arc
 120 INPUT ra: REM arc radius
 130 INPUT ds: REM dot spacing
 140 INPUT as,ae: REM start and
end angles for arc
 195 REM
 200 REM draw arc
 205 REM
 210 LET ds=ds*3.14159/180
 220 LET as=as*3.14159/180
 230 LET ae=ae*3.14159/180
 240 LET r=ra
 250 FOR p=as TO ae STEP ds
 260 LET x=r*COS (p)
 270 LET y=r*SIN (p)
 280 LET x=x+xc
 290 LET y=y+yc
 300 PLOT x,y
 310 NEXT p
 320 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

DISK 1

DESCRIPTION

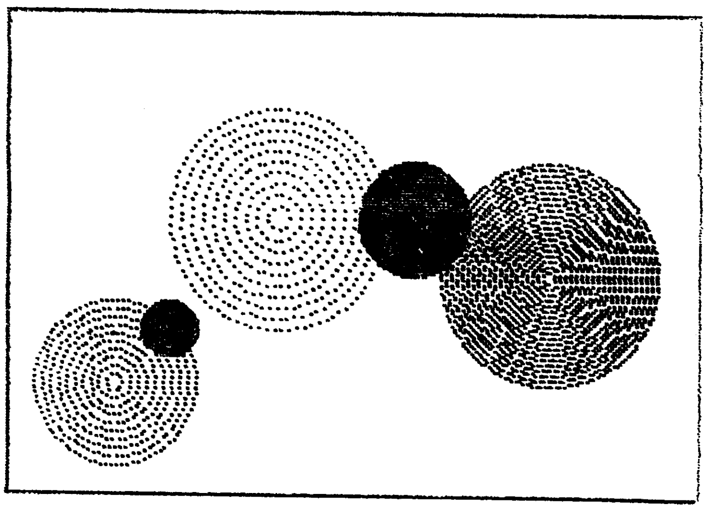
When examining the program CIRCLE, you probably realised that if you repeated the process again and again, but specifying a different radius each time, it would be possible to build up a complete disk rather than just a circle. This is certainly true, but the time taken would be rather a long one, and you'd probably get fed up with running through the program time after time. Consequently, the program DISK 1 takes the drudgery out of the process by incorporating a couple of new routines to do it all for you.

RUNNING THE PROGRAM

Again, we have to input a number of variables before we get to the meat of the program. As before, line 110 allows us to specify the coordinates of the disk centre, line 120 the disk radius, and line 130 the dot spacing. In drawing the disk however, we go through two FOR NEXT loops rather than the usual one. The inner loop, lines 230 to 290, draws just one circle as we've seen before. The loop in line 220 and 300 then uses the previously specified dot separation to step up the radius of the circle to draw another one, until finally we reach the full radius originally input in line 120.

PROGRAM STRUCTURE

60-70	set colours
90	draw border round screen using subroutine at 400
110	input coordinates of disk centre
120	input disk radius
130	input dot separation
210-290	draw arc, incorporating:-
230-290	draw circle, and
220,300	step up radius and draw another one
320	back for another go
400-460	border drawing routine



```
   1 REM DISK 1
   2 REM ************************
   3 REM
  10 REM routine to draw a disk
  20 REM spacing between the dot
s used
  30 REM to draw the disk is var
iable
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM input disk drawing para
meters
 110 INPUT xc,yc: REM coordinate
s of disk centre
 120 INPUT ra: REM disk radius
 130 INPUT ds: REM dot spacing
 195 REM
 200 REM draw disk
 205 REM
 210 LET d=ds*3.14159/180
 220 FOR r=ds TO ra STEP ds
 225 REM
 230 FOR p=0 TO 2*3.14159 STEP d
*(40/r)
 240 LET x=r*COS (p)
 250 LET y=r*SIN (p)
 260 LET x=x+xc
 270 LET y=y+yc
 280 PLOT x,y
 290 NEXT p
 300 NEXT r
 310 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

DISK 2

DESCRIPTION

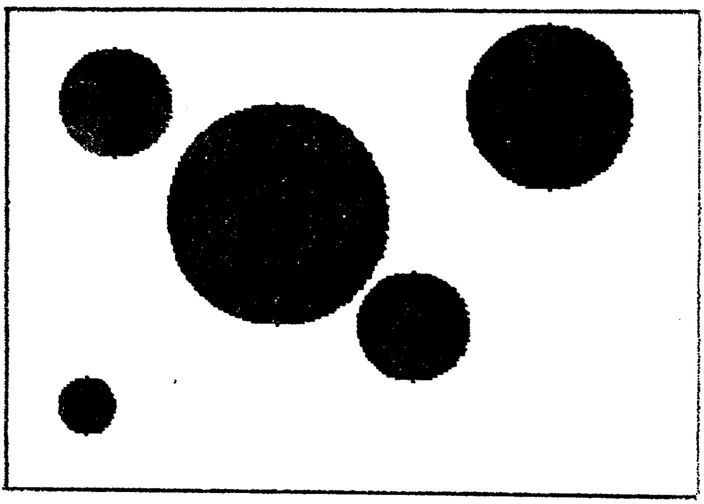
In the previous program Disk 1 a disk was constructed by repeatedly drawing a circle of ever increasing radius centred around the same spot. The one disadvantage of this is that, to draw a solid disk, we have to draw an awful lot of circles, and this of course takes quite a long time. In order to speed up the process the following program introduces the DRAW command. As you may know, DRAW takes as its starting point the last point to be drawn using any one of the three commands PLOT, CIRCLE, and DRAW itself. There is a third option, that of specifying an angle to be drawn through, but that need not concern us here. By plotting points on the circumference of a circle, we can use the DRAW command to draw a line from that point to A point on the other side of the circle that has the same Y coordinate. Thus the disk is built up from bottom to top by a series of lines.

RUNNING THE PROGRAM

The only inputs required in this program are the central X, Y coordinates for the disk (XC and YC), and the radius RA. In lines 210 and 220 we calculate the start and end Y coordinates: in other words the bottom and top of the disk. Lines 240 to 270 then calculate the starting coordinates for the PLOT and DRAW commands, namely XS, and in order to be able to use DRAW we also work out the end X coordinate XE. Thus, in line 280 we PLOT a point on the left hand side of the disk, and in 290 DRAW a horizontal line over to the right hand edge. Note that line 290 should read DRAW XE, Y, and NOT DRAW XE, 0. You can always try 0 and see what happens!

PROGRAM STRUCTURE

60-70	set colours
80	draw border using routine at 400
110	input central coordinates of disk
120	input radius of disk
210-220	calculate start and end Y coordinates
240-270	calculate start and end X coordinates
280-290	PLOT and DRAW the horizontal line
310	back for another go
400-460	border drawing subroutine



```
   1 REM DISK 2
   2 REM ************************
   3 REM
  10 REM routine to draw a solid
 disk
  20 REM using the DRAW command
to increase
  30 REM the disk drawing speed
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM input disk drawing para
meters
 110 INPUT xc,yc: REM coordinate
s of disk centre
 120 INPUT ra: REM disk radius
 195 REM
 200 REM draw disk
 205 REM
 210 LET ys=yc-ra
 220 LET ye=yc+ra
 225 REM
 230 FOR y=ys TO ye
 240 LET c=y-yc
 250 LET l=SQR (ra*ra-c*c)
 260 LET xs=xc-l
 270 LET xe=(xc+l)-xs
 280 PLOT xs,y
 290 DRAW xe,0
 300 NEXT y
 310 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

SEGMENT

DESCRIPTION

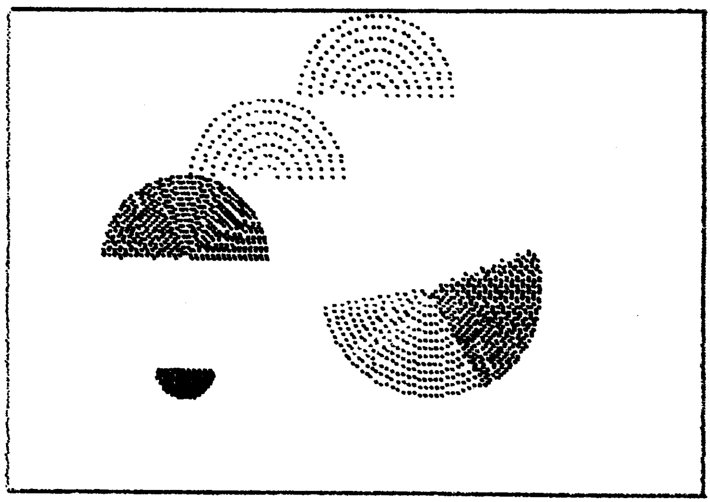
Although DRAW allows one to draw an arc, it does not allow one to draw an arc with variable dot spacing. By drawing various circles to variable dot spacing, a disk with the same dot spacing can be plotted. Combining both of these routines resulted in the program Segment, presented here. Using this program we can draw a disk segment, again with the spacing between the dots defined by an input (line 130), and moreover we can make that segment as large, or as small, as we like. As you can see from the illustration, combining a number of runs of the program enables us to link different disk segments together.

RUNNING THE PROGRAM

As usual, line 110 lets us input the coordinates of the arc centre, 120 the arc radius, and 130 the spacing between the dots. In line 140 we input the start and end angles for the arc. The program following is then fairly straightforward. In lines 250 to 310 we plot just one arc, using the PLOT command for each point of the arc. The outer FOR NEXT loop, in lines 240 and 320, uses the dot separation to increase the radius of the arc, and then the inner loop plots out another arc. This continues until we reach the final radius of the arc, RA, as input in line 120, which gives us our final arc and completes the segment. By specifying a different dot spacing we can build up a whole series of arcs joined onto each other.

PROGRAM STRUCTURE

60-70	set colours
90	draw border using routine at 400
110	input central coordinates of arc
120	input radius of arc
130	input dot spacing
140	input start and end angles for arc
240,320	outer drawing routine, incorporating:
250-310	individual arc drawing routine
330	back for another go
400-460	border drawing subroutine



```
   1 REM SEGMENT
   2 REM ***********************
   3 REM
  10 REM routine to draw a disk
segment
  20 REM spacing between the dot
s used
  30 REM to draw the segment is
variable
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM input arc drawing param
eters
 110 INPUT xc,yc: REM coordinate
s of arc centre
 120 INPUT ra: REM arc radius
 130 INPUT ds: REM dot spacing
 140 INPUT as,ae: REM start and
end angles for arc
 195 REM
 200 REM draw arc
 205 REM
 210 LET d=ds*3.14159/180
 220 LET as=as*3.14159/180
 230 LET ae=ae*3.14159/180
 240 FOR r=ds TO ra STEP ds
 250 FOR p=as TO ae STEP d*(40/r
)
 260 LET x=r*COS (p)
 270 LET y=r*SIN (p)
 280 LET x=x+xc
 290 LET y=y+yc
 300 PLOT x,y
 310 NEXT p
 320 NEXT r
 330 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

PIECHART

DESCRIPTION

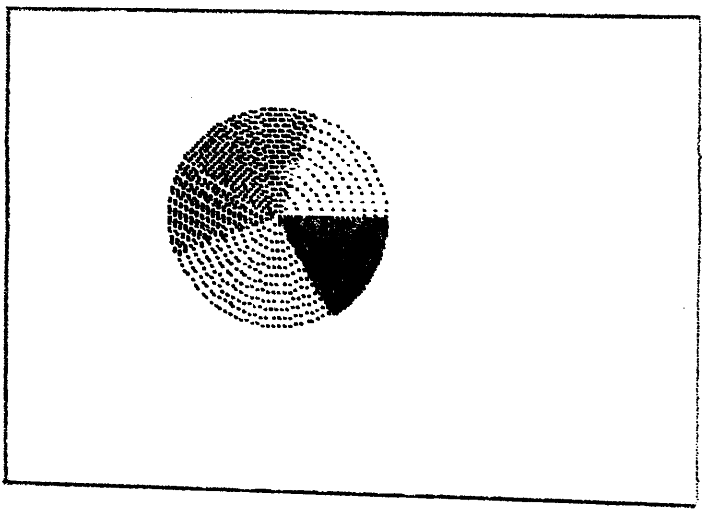
The culmination of all the plotting routines for circles, arcs, and disks results in the program PIECHART. Of use in business, educational, and indeed just about any computing environment, piecharts enable us to show clearly and (quite strikingly) visually all manner of different data. We mentioned when describing the program Segment, that by building up various runs through the program it was possible to have different segments next to each other. This program takes the chore out of that exercise, by assigning various variables first of all, and then using DATA statements to generate the necessary information. Obviously, this program will be of most use to you when using your own data.

RUNNING THE PROGRAM

This program differs from the earlier Segment one by having no input statements. Instead, we define the variables XC and YC to be the central coordinates in line 110, and the variable RA to be the radius in line 120. Needless to say you can change these to suit your own requirements. The data for making up the different arc segments is contained in lines 500 to 560. In order, we have the dot separation, the start angle for the segment, and the end angle. Again, these can be whatever you require. By reading these in lines 130 and 140, we then follow the segment plotting routine in lines 240 to 320. When line 150 detects a zero dot separation (as read in from line 540) the program comes to a halt.

PROGRAM STRUCTURE

60-70	set colours
90	draw border using routine at 400
110	define coordinates of centre of piechart
120	define radius of piechart
130	READ dot spacing
140	READ start and end angles of segment
150	if spacing of zero, then STOP
210-320	segment drawing routine
330	back for more data
400-460	border drawing subroutine
500-540	data for piechart



```
   1 REM PIECHART
   2 REM ************************
   3 REM
  10 REM routine to draw a piech
art using
  20 REM variable spacing betwee
n the dots
  30 REM to differentiate betwee
n segments.
  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  90 GO SUB 400
  95 REM
 100 REM get piechart data from
data tables - line 500+
 105 REM
 110 LET xc=100: LET yc=100: REM
 coordinates of disk centre
 120 LET ra=40: REM disk radius
 130 READ ds: REM dot spacing
 140 READ as,ae: REM start and e
nd angles for segment
 150 IF ds=0 THEN STOP
 195 REM
 200 REM draw segment
 205 REM
 210 LET d=ds*3.14159/180
 220 LET as=as*3.14159/180
 230 LET ae=ae*3.14159/180
 240 FOR r=ds TO ra STEP ds
 250 FOR p=as TO ae STEP d*(40/r
)
 260 LET x=r*COS (p)
 270 LET y=r*SIN (p)
 280 LET x=x+xc
 290 LET y=y+yc
 300 PLOT x,y
 310 NEXT p
 320 NEXT r
 330 GO TO 100
 395 REM
 400 REM border drawing routine
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
 495 REM
 500 DATA 4,1,70
 510 DATA 2,71,200
 520 DATA 3,201,300
 530 DATA 1,301,360
 540 DATA 0,0,0
```

GRAPH PLOTTING

GRAPH

DESCRIPTION

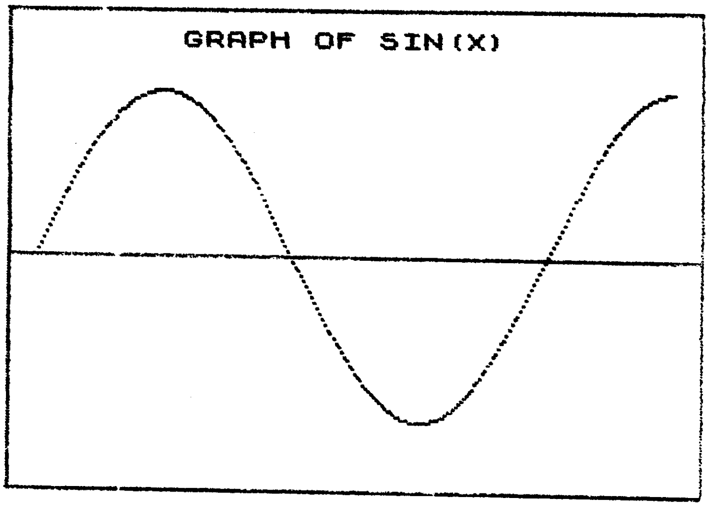
Your computer can plot points on a screen that is 32 columns across and 22 rows deep, giving us a total of 32 x 24 or 704 character positions. Each of these character positions is made up of an 8 x 8 dot matrix, which means that we can plot points to a resolution of 256 in the X axis and 176 in the Y axis. The two programs GRAPH and GRAPH2 use the full resolution of the screen to plot respectively a graph of SIN (X) and SIN (X) with COS (X), using the computer commands PLOT, and DEF FN to define the function to be plotted. The programs are identical except for an additional routine in GRAPH2 to plot COS (X), and a couple of lines to identify the function and display a title.

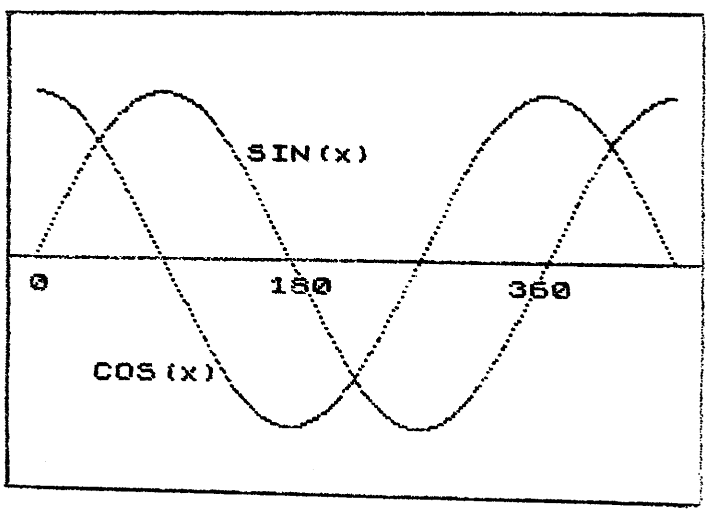
RUNNING THE PROGRAM

The INPUTting of variables is not required in either program, as we are simply taking the function a(x) to represent sin (x/30)*60 in the program GRAPH, and in addition b(x) to represent cos (x/30)*60 in the program GRAPH2. These are defined in line 130 in the former program, and lines 130-131 in the latter. It then runs through lines 200 to 390 to plot out the actual function. These routines could obviously be incorporated in further programs to plot different functions, just by altering the definitions in lines 130-131.

PROGRAM STRUCTURE

60-70 set colours
90 draw border around screen using subroutine at 400
130-131 define function(s) to be plotted
150-180 draw Y axis and label graph(s)
200-390 graph plotting routine
400-460 border drawing subroutine


GRAPH OF SIN(X)


SIN(x)
0
180
360
COS(x)

```
   1 REM GRAPH
   2 REM ************************
   3 REM
  10 REM program to plot the gra
ph of a function
  20 REM
  30 REM set colours
  40 REM
  50 INK 0
  60 PAPER 7
  70 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM define function to be p
lotted
 105 REM
 110 REM the numerical values in
 the example are
 120 REM used to scale the plot
to reasonable dimensions
 125 REM
 130 DEF FN a(x)=SIN (x/30)*60
 135 REM
 140 REM draw Y axis at 0
 145 REM
 150 PLOT 1,85
 160 DRAW 254,0
 180 PRINT AT 1,8;"GRAPH OF SIN(
X)"
 185 REM
 190 REM plot graph
 195 REM
 200 FOR x=1 TO 235
 210 LET y=FN a(x)
 220 PLOT x+10,y+85
 230 NEXT x
 390 STOP
 395 REM
 400 REM draw border around scre
en
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

```
   1 REM GRAPH 2
   2 REM ***********************
   3 REM
  10 REM program to plot the gra
ph of two functions
  20 REM
  30 REM set colours
  40 REM
  50 INK 0
  60 PAPER 7
  70 REM
  80 REM draw border
  85 REM
  90 GO SUB 400
  95 REM
 100 REM define function to be p
lotted
 105 REM
 110 REM the numerical values in
the example are
 120 REM used to scale the plot
to reasonable dimensions
 125 REM
 130 DEF FN a(x)=SIN (x/30)*60
 131 DEF FN b(x)=COS (x/30)*60
 135 REM
 140 REM draw Y axis and labels
 145 REM
 150 PLOT 1,85
 160 DRAW 254,0
 165 PRINT AT 12,1;"0            1
80          360"
 170 PRINT AT 6,11;"SIN(x)"
 180 PRINT AT 16,4;"COS(x)"
 185 REM
 190 REM plot graph
 195 REM
 200 FOR x=1 TO 235
 210 LET y=FN a(x)
 220 PLOT x+10,y+85
 230 LET y=FN b(x)
 240 PLOT x+10,y+85
 300 NEXT x
 390 STOP
 395 REM
 400 REM draw border around scre
en
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

3D GRAPH

DESCRIPTION

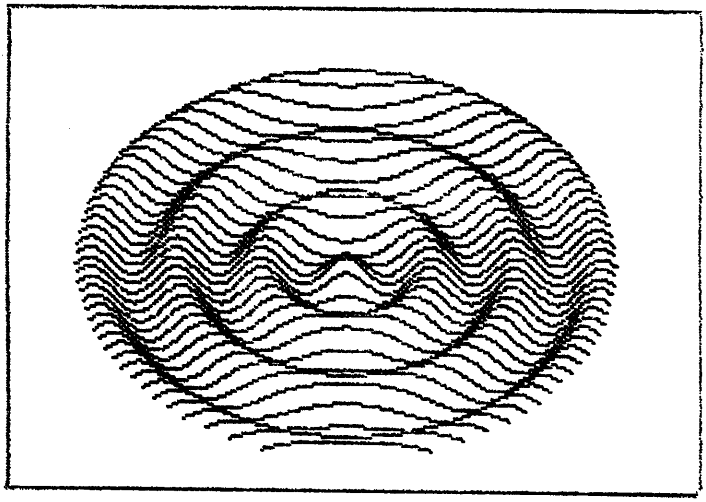
Building on from the routines for plotting two dimensional functions, we find that it is relatively easy to design a program for plotting in three dimensions. The two programs labelled 3D Graph do just that. Although we are relying on the same computer command PLOT, our routine for plotting the function is, of necessity, rather more complicated this time, as we are trying to emulate a three dimensional image on what is, after all, a two dimensional screen. Of special interest in this routine is the double IF statement in line 310, which performs a straightforward RETURN depending on the values of the variables P and Z. Two versions of the program are given: these are identical apart from the definition of the function to be plotted, which is in line 150.

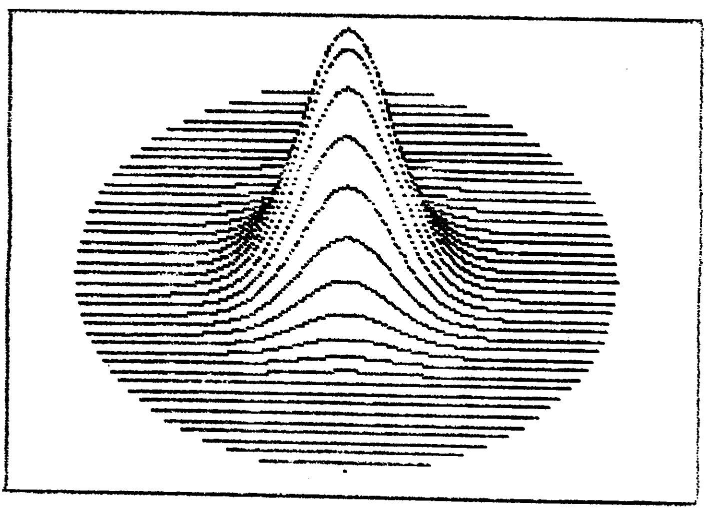
RUNNING THE PROGRAM

No variables are INPUT in this program, as our function is defined in line 150, and the area to be plotted in is determined by the scale given to X in line 220. This in turn determines the scale of Y to be plotted, by line 260. Line 270, the start of the inner of our two plotting loops, plots all the points on the Y axis for the value of X in the outer loop, which commences at line 220. We then move onto the next point on the X axis, and plot all the Y values there, and so on. By changing the definition in line 150 we can plot out a whole series of different functions.

PROGRAM STRUCTURE

60-70	set colours
90	draw border round screen using subroutine at 400
150	define function to be plotted
210-380	plotting routine
370-390	program termination
400-460	border drawing subroutine





```
   1 REM 3D GRAPH
   2 REM ***********************
   3 REM
  10 REM this routine plots the
graph of a
  20 REM function in 3 dimension
s
  30 REM
  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  80 REM draw border around scre
en
  90 GO SUB 400
  95 REM
 100 REM define function to be p
lotted
 105 REM
 110 REM the function is changed
 by altering the contents of lin
e 150
 120 REM
 150 DEF FN a(z)=90*EXP (-z*z/60
0)
 195 REM
 200 REM plot graph
 205 REM
 210 LET k=5
 220 FOR x=-100 TO 100 STEP 1
 230 LET l=0
 240 LET p=1
 250 LET z1=0
 260 LET y1=k*INT (SQR (10000-x*
x)/k)
 270 FOR y=y1 TO -y1 STEP -k
 280 LET z=INT (80+FN a(SQR (x*x
+y*y))-.707106*y)
 290 IF z<l THEN GO TO 350
 295 GO SUB 380
 300 LET l=z
 310 IF p=0 THEN GO SUB 380: IF
z=z1 THEN GO SUB 380
 320 PLOT x+125,z
 330 IF p=0 THEN LET z1=z
 340 LET p=0
 350 NEXT y
 360 NEXT x
 370 GO SUB 390
 380 RETURN
 390 STOP
 395 REM
 400 REM draw border
 405 REM
 410 PLOT 0,0
 420 DRAW 255,0
 430 DRAW 0,175
 440 DRAW -255,0
 450 DRAW 0,-175
 460 RETURN
```

INTERPOLATE

DESCRIPTION

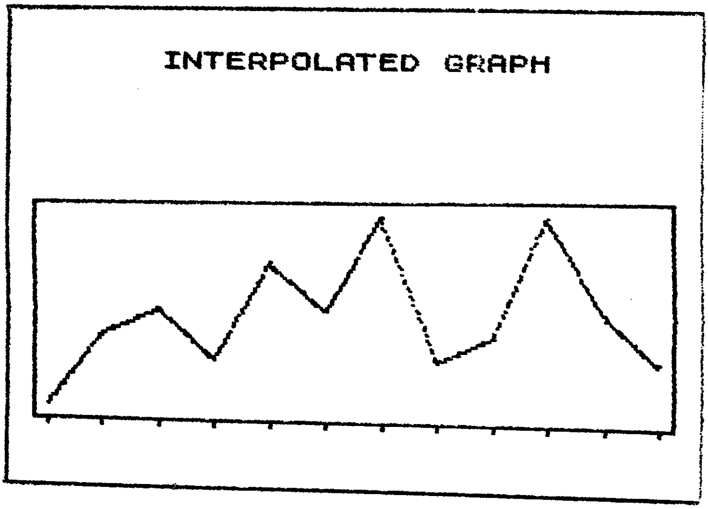
Determining a set of data is all very well, but it is the interpolation of that data that produces the all important results. One common method of doing this is to take the data and turn it into points on a graph, and then perform the interpolation between those points. The program "Interpolate" does that, by assuming that you already have your data in the form of X,Y co-ordinates (here we store them as data statements in line 180), and plotting the appropriate point out within a defined area (lines 220 to 230 define the top and bottom of the Y axis and left and right of the X axis), before finally 'joining up' the points in whatever form you desire (see Running the Program). You could quite easily incorporate your own data into this program simply by changing the data statements in line 180.

RUNNING THE PROGRAM

The main bulk of the work is done a) by the line 180, which stores the data as X,Y co-ordinates, and b) line 200, which determines which point we start at (here it is the first one), which one we finish at (here it is the twelfth), and which points we interpolate between (here it is every one), although by changing the variable SP in line 200 we could easily take every other point, for instance. Once we've calculated the scaling factors in lines 410 to 490, and turned these into point increments in lines 510 and 520, we plot the actual point in line 640, and the line between each point by the routine in lines 670 - 730.

PROGRAM STRUCTURE

60 - 70	set colours
90	draw border round screen using subroutine at 1000
110 - 160	read and store the data
180	data stored as X,Y co-ordinate
200	determine start and finish points, and separation
220 - 230	determine position and dimensions of graph on screen
310 - 380	draw border round graph, and label graph
410 - 490	determine scaling factors
510 - 520	convert scaling factors to point increments
610 - 740	point and line drawing routine
1000 - 1060	border drawing routine


INTERPOLATED GRAPH

```
   1 REM INTERPOLATE
   2 REM *********************
   3 REM
  10 REM program to draw a graph
 by interpolating
  20 REM a set of points stored
as data statements in
  30 REM line 180.
  45 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 1000
  95 REM
 100 REM initialise data
 105 REM
 110 DIM x(12)
 120 DIM y(12)
 130 FOR i=1 TO 12
 140 READ x(i)
 150 READ y(i)
 160 NEXT i
 165 REM
 170 REM data stored as X and Y
coordinate
 175 REM
 180 DATA 1,10,2,25,3,30,4,20,5,
40,6,30,7,50,8,20,9,25,10,50,11,
30,12,20
 185 REM
 190 REM min dimension =1, max=1
2, seperation =1
 195 REM
 200 LET dn=1: LET dx=12: LET sp
=1
 205 REM
 210 REM position and dimensions
 of graph on screen
 215 REM
 220 LET xl=240: LET xr=15
 230 LET yt=100: LET yb=30
 295 REM
 300 REM draw border around grap
h
 305 REM
 310 PLOT xr-5,yb-5
 320 DRAW (xl-xr+10),0
 330 DRAW 0,(yt-yb+10)
 340 DRAW -(xl-xr+10),0
 350 DRAW 0,-(yt-yb+10)
 355 LET xi=(xl-xr)/(dx-dn)
 360 FOR x=xr TO xl STEP xi
 365 PLOT x,yb-6: PLOT x,yb-7
 370 NEXT x
```

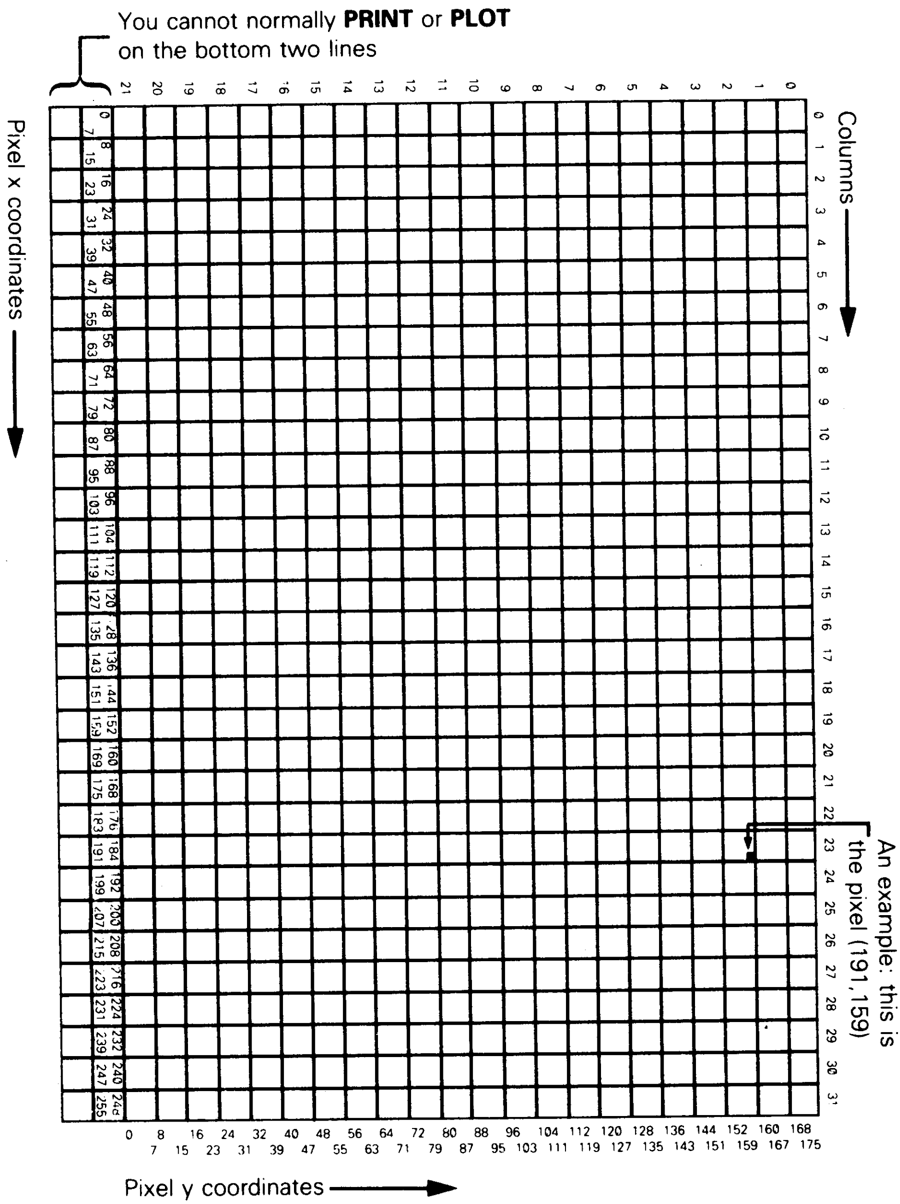
```
 380 PRINT AT 2,7;"INTERPOLATED
GRAPH"
 395 REM
 400 REM calculate scale factors
 405 REM
 410 LET y1=-1000000
 420 LET y2=1000000
 430 LET x1=y1: LET x2=y2
 440 FOR i=dn TO dx STEP sp
 450 IF y1<y(i) THEN LET y1=y(i)
 460 IF y2>y(i) THEN LET y2=y(i)
 470 IF x1<x(i) THEN LET x1=x(i)
 480 IF x2>x(i) THEN LET x2=x(i)
 490 NEXT i
 495 REM
 500 REM convert scaling factors
into point increments
 505 REM
 510 LET a=(x1-x2)/(xl-xr)
 520 LET b=(y1-y2)/(yt-yb)
 595 REM
 600 REM plot graph
 605 REM
 610 FOR i=dn TO dx STEP sp
 620 LET x=(xr+(x(i)-x2)/a)
 630 LET y=((y(i)-y2)/b+yb)
 640 PLOT x,y
 650 LET q=i+sp
 660 IF q>dx THEN STOP
 670 FOR j=x(i) TO x(q) STEP .03
 680 LET y3=((y(q)-y(i))/(x(q)-x
(i)))*(j-x(i))+y(i)
 690 LET x=INT (xr+(j-x2)/a)
 700 LET y=INT ((y3-y2)/b+yb)
 710 PLOT x,y
 730 NEXT j
 740 NEXT i
1000 REM draw border around scre
en
1005 REM
1010 PLOT 0,0
1020 DRAW 255,0
1030 DRAW 0,175
1040 DRAW -255,0
1050 DRAW 0,-175
1060 RETURN
```

USING THE VIDEO MEMORY

GRAPHICS CHARACTERS

There are sixteen special graphics symbols which can be displayed on your computer. They are what are known as quarter square graphics; this means that each character space is divided into four and the symbols are built up from one or more of the quarters being in the ink colour. These symbols are shown on the keyboard and are accessible by first pressing the key marked GRAPHICS (note that the cursor letter changes to 'G') then pressing one of the keys bearing the appropriate graphics symbol. There are only eight keys bearing the graphics symbol legend, the remaining eight symbols can be obtained by pressing the shift key as well as the symbol key.

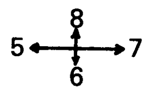
In addition to the sixteen quarter square graphics symbols there is the facility to create up to 21 user defined graphics. As with the quarter square symbols these are displayed using the GRAPHICS key followed by one of the keys from A to U. When the machine is first switched on these will contain just the corresponding alphabetical letters. To get a user defined character it must first be designed and entered into the correct eight bytes of a special 168 byte RAM character generator at the top of memory. The best way of designing the character is to use a character editor – such a program is included in this section. The character editor produces a set of eight values which determine the pattern of the character. If these values are converted into binary form then the '1' values indicate that the pixel is in ink colour and a '0' that the pixel is in paper colour. Having obtained the eight values for each character these values can be placed in the RAM character generator, and used by another program.


You cannot normally PRINT or PLOT
on the bottom two lines
Columns
Pixel x coordinates
Pixel y coordinates
An example: this is
the pixel (191,159)

HI-RES CURSOR

DESCRIPTION

Many of the arcade games about at present require the movement of some kind of 'sight' around the screen, to get you to the right position before firing. Similarly, a routine to move a sight, or indeed a cursor over the screen, would have many uses in plotting, design, and graphic programs generally. The two programs here provide just such a routine, but achieved in slightly different manners. What they do have in common is the method of moving the cursor (here it is a cross) about, which uses the keys 5, 6, 7 and 8 in the following way:



Thus, pressing the 8 key would move the cursor up, etc. This routine lies in lines 210 to 260. The two programs differ in that a) the cursor is designed differently in each one, and b) the first program erases whatever screen contents the cursor passes over: the second one doesn't.

RUNNING THE PROGRAM HI-RES CURSOR

Having drawn our border around the screen, the program positions the cursor at the X, Y coordinate of 5,5; and sets the increment between cursor movements (the variable S in line 120) to be 4. The program then simply waits until you press the appropriate key, increases or decreases X or Y accordingly, and then checks to see whether you are still within the screen boundary. If you are, it erases the previous cursor, draws a new one, and then awaits the pressing of another key.

PROGRAM STRUCTURE FOR HI-RES CURSOR

60-70 set colour
90 draw border round screen using subroutine at 600
110-120 set up parameters
210-260 check for key press
310-340 check if within boundary
410-440 erase previous cursor
460-490 draw new cursor
510 back to check for key press

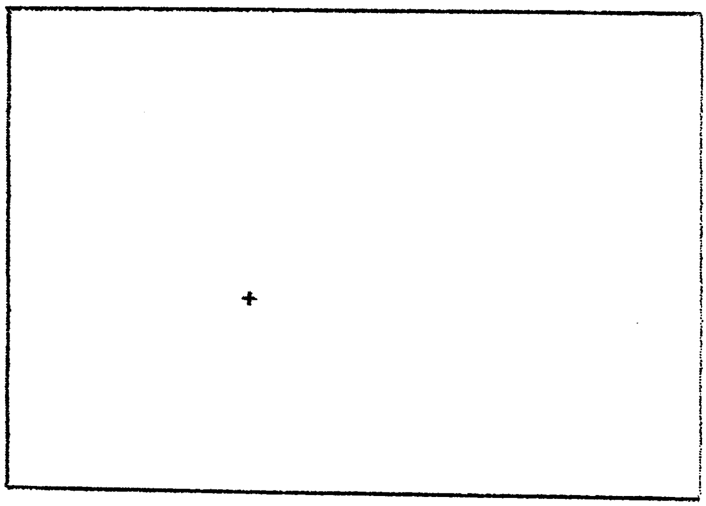
600-660 border drawing subroutine

RUNNING THE PROGRAM HI-RES CURSOR 2

This follows roughly the same lines as Hi-Res Cursor, although our cursor is now defined in data statements contained in lines 150-170, and stored in the array C (5,5). This 'cursor' can now be anything you like, simply by changing the data statements. We plot the position of the cursor in lines 610-680, using the computer command INVERSE. In other words, we replace paint dots by ink dots, and vice versa. However, the point of this program is that we do NOT erase the screen contents, so the routine from lines 410 to 480 erases the cursor but then does an INVERSE on what has just gone, thus restoring the original screen display. Before plotting the cursor again we must save the screen contents into our array M(5,5), and this is performed by the function in lines 510 to 550, using the computer command POINT. POINT tells us whether a pixel is paper or ink colour, and we use this array again when going back to erase the cursor and re-trace the screen contents.

PROGRAM STRUCTURE FOR HI-RES CURSOR 2

60-70 set colours
90 fill the entire screen with characters
105-110 set up parameters
120-170 define 'cursor' and read data statements
210-260 wait for appropriate key press
310-345 check if within boundary
410-480 erase previous cursor and restore screen contents
510-550 save screen contents
610-680 plot new cursor
910 go back and wait for another key to be pressed



```
   1 REM HI-RES CURSOR
   2 REM ***********************
   3 REM
  10 REM program to move a high
resolution cursor about
  20 REM the screen under contro
l of the keyboard.
  30 REM
  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 600
  95 REM
 100 REM set up parameters
 110 LET x=5: LET y=5: REM start
 position
 120 LET s=4: REM cursor movemen
t increments
 130 GO TO 450
 195 REM
 200 REM input cursor movement f
rom keyboard
 205 REM
 210 IF INKEY$="" THEN GO TO 210
 220 LET xo=x: LET yo=y
 230 IF INKEY$="5" THEN LET x=x-
s
 240 IF INKEY$="6" THEN LET y=y-
s
 250 IF INKEY$="7" THEN LET y=y+
s
 260 IF INKEY$="8" THEN LET x=x+
s
 295 REM
 300 REM check cursor within bou
nds
 305 REM
 310 IF x<5 THEN LET x=5
 320 IF x>250 THEN LET x=250
 330 IF y<5 THEN LET y=5
 340 IF y>170 THEN LET y=170
 395 REM
 400 REM erase previous cursor
 405 REM
 410 PLOT INVERSE 1;xo-2,yo
 420 DRAW INVERSE 1;4,0
 430 PLOT INVERSE 1;xo,yo-2
 440 DRAW INVERSE 1;0,4
 445 REM
 450 REM plot new cursor
 455 REM
 460 PLOT x-2,y
 470 DRAW 4,0
 480 PLOT x,y-2
```

```
490 DRAW 0,4
495 REM
500 REM do again
505 REM
510 GO TO 210
595 REM
600 REM draw border around scre
en
605 REM
610 PLOT 0,0
620 DRAW 255,0
630 DRAW 0,175
640 DRAW -255,0
650 DRAW 0,-175
660 RETURN
```

```
  1 REM HI-RES CURSOR 2
  2 REM ***********************
  3 REM
 10 REM program to move a high
resolution cursor about
 20 REM the screen under contro
l of the keyboard.
 30 REM the cursor does not era
se existing screen displays
 40 REM
 50 REM set colours
 60 INK 0
 70 PAPER 7
 75 REM
 80 REM fill screen with charac
ters
 85 REM
 90 FOR q=1 TO 704: PRINT "*";:
 NEXT q
 95 REM
100 REM set up parameters
105 LET x=5: LET y=5: REM start
 position
110 LET s=4: REM cursor movemen
t increments
120 DIM c(5,5): DIM m(5,5)
125 FOR i=1 TO 5
130 FOR j=1 TO 5
135 READ c(j,i)
140 NEXT j
145 NEXT i
150 DATA 0,0,1,0,0
155 DATA 0,0,1,0,0
160 DATA 1,1,1,1,1
165 DATA 0,0,1,0,0
170 DATA 0,0,1,0,0
190 GO TO 500
```

```
 195 REM
 200 REM input cursor movement f
rom keyboard
 205 REM
 210 IF INKEY$="" THEN GO TO 210
 220 LET xo=x: LET yo=y
 230 IF INKEY$="5" THEN LET x=x-
s
 240 IF INKEY$="6" THEN LET y=y-
s
 250 IF INKEY$="7" THEN LET y=y+
s
 260 IF INKEY$="8" THEN LET x=x+
s
 295 REM
 300 REM check cursor within bou
nds
 305 REM
 310 IF x<5 THEN LET x=5
 320 IF x>250 THEN LET x=250
 330 IF y<5 THEN LET y=5
 340 IF y>170 THEN LET y=170
 395 REM
 400 REM erase previous cursor
 405 REM
 410 FOR i=-2 TO 2
 420 FOR j=-2 TO 2
 430 IF m(j+3,i+3)=0 THEN GO TO
460
 440 PLOT j+xo,i+yo
 450 GO TO 470
 460 PLOT INVERSE 1;j+xo,i+yo
 470 NEXT j
 480 NEXT i
 495 REM
 500 REM save screen contents
 505 REM
 510 FOR i=-2 TO 2
 520 FOR j=-2 TO 2
 530 LET m(j+3,i+3)=POINT (j+x,i
+y)
 540 NEXT j
 550 NEXT i
 595 REM
 600 REM plot new cursor
 605 REM
 610 FOR i=-2 TO 2
 620 FOR j=-2 TO 2
 630 IF c(j+3,i+3)=0 THEN GO TO
660
 640 PLOT j+x,i+y
 650 GO TO 670
 660 PLOT INVERSE 1;j+x,i+y
 670 NEXT j
 680 NEXT i
 895 REM
 900 REM do again
 905 REM
 910 GO TO 210
```

CHARACTER EDITOR

DESCRIPTION

To use the high resolution capabilities of the computer to the full, we need to know a little bit about the commands POKE, USR and CHR$. You'll find the relevant information in your Timex-Sinclair Basic Programming book. Quickly, we can use CHR$(144) to CHR$(164) for storing your own high resolution characters, and later these can be accessed from the computer keyboard by going into graphics mode and pressing any of the keys A through U. By using the program Character Editor we can define our own character (character 1 will later be the letter A, character 2 will be B, and so on), and you'll see a display of 8 numerical values on the screen when this is done. In the program, line 1710 uses the USR function to return the address of the first byte in memory for the user defined character which we want to be represented by whatever letter: if we want A to represent a character, in that line C would equal zero. Study both the listing and the section on Running the Program carefully. Lines 1720 to 1740 then POKE the aforementioned 8 numerical values into memory, so that (until we reset or turn the machine off), our new character is stored.

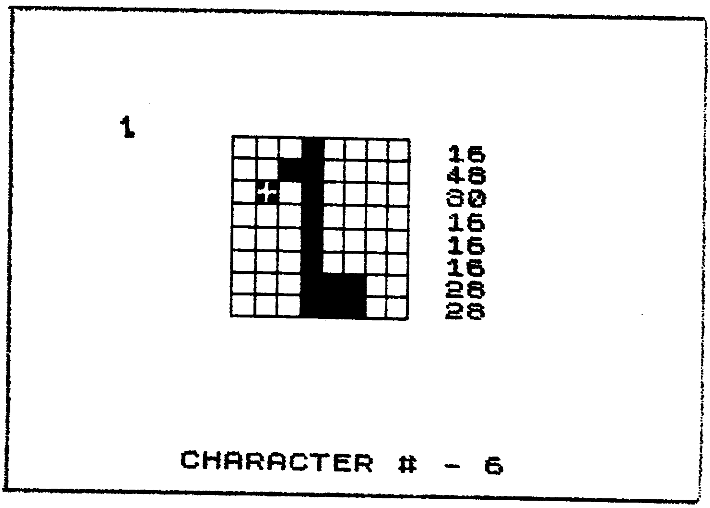
RUNNING THE PROGRAM

After INPUTting a character number C, which can lie in the range 0 through 20, a border is drawn around the screen and the display grid drawn in the middle. Using previous routines a high resolution cursor is displayed, and, again as before, pressing keys 5, 6, 7 or 8 moves the cursor around the grid. At any of the squares on the grid, pressing A will add a point to the display, pressing D will delete a point, and pressing N will start everything up again. At any time when you've pressed either A or D, our character data values are updated, and a 'life size' representation of the character displayed on the screen, by the line 1710 to 1800.

PROGRAM STRUCTURE

60-70	set colours
125	draw border round screen using subroutine at 2000
130-150	input character number
160	draw border round screen using subroutine at 2000
280	draw grid using subroutine at 1300
290	position cursor by subroutine at 555
310-392	await appropriate key press

410-445	check cursor within grid
510-550	draw cursor
710-730	add point to character
740	recalculate character value, print them out again and redraw character using subroutine at 1500
910-930	delete point from character
940	recalculate character value, print them out again and redraw character using subroutine at 1500
1310-1450	display grid, character values and character numbers
1510-1680	calculate character values and display them on screen
1710-1750	POKE new values into memory and display character on screen
2000-2060	border drawing subroutine


1
16
48
80
16
16
16
28
28
CHARACTER # - 6

```
   1 REM CHARACTER EDITOR
   2 REM ***********************
   3 REM
  10 REM this program allows the
 easy creation and display
  20 REM of user definable chara
cters.
  30 REM
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  85 REM
 100 REM input and set up parame
ters
 105 REM
 110 LET x=10: LET y=6
 120 DIM c(8)
 125 CLS : GO SUB 2000
 130 PRINT AT 20,6;"Input charac
ter number ";
 140 INPUT c: PRINT c
 150 CLS
 160 GO SUB 2000
 280 GO SUB 1300
 290 GO TO 550
 300 REM keyboard input
 305 REM
 310 IF INKEY$="" THEN GO TO 310
 320 LET xo=x: LET yo=y
 330 IF INKEY$="a" THEN GO TO 70
0
 340 IF INKEY$="d" THEN GO TO 90
0
 350 IF INKEY$="5" THEN LET x=x-
1: GO TO 400
 360 IF INKEY$="6" THEN LET y=y+
1: GO TO 400
 370 IF INKEY$="7" THEN LET y=y-
1: GO TO 400
 380 IF INKEY$="8" THEN LET x=x+
1: GO TO 400
 390 IF INKEY$="n" THEN GO TO 10
0
 392 GO TO 300
 395 REM
 400 REM check cursor within gri
d
 405 REM
 410 IF x<10 THEN LET x=10
 420 IF x>17 THEN LET x=17
 430 IF y<6 THEN LET y=6
 440 IF y>13 THEN LET y=13
 495 REM
 500 REM draw cursor
 505 REM
 510 LET xc=xo*8: LET yc=(21-yo)
*8
```

```
 520 PLOT INVERSE 1;xc+2,yc+4
 530 DRAW INVERSE 1;4,0
 540 PLOT INVERSE 1;xc+4,yc+2
 550 DRAW INVERSE 1;0,4
 555 REM
 560 LET xc=x*8: LET yc=(21-y)*8
 570 PLOT xc+2,yc+4
 580 DRAW 4,0
 590 PLOT xc+4,yc+2
 600 DRAW 0,4
 610 GO TO 300
 695 REM
 700 REM add point to character
 705 REM
 710 LET q=x+22528+(32*y)
 715 IF PEEK (q)=10 THEN GO TO 3
00
 716 IF PEEK (q)=8 THEN GO TO 30
0
 720 POKE q,10
 730 LET p=0
 740 GO SUB 1500
 800 GO TO 300
 900 REM delete point from chara
cter
 910 LET q=x+22528+(32*y)
 915 IF PEEK (q)=56 THEN GO TO 3
00
 920 POKE q,56
 930 LET p=1
 940 GO SUB 1500
 950 GO TO 300
1300 REM display grid
1310 FOR g=64 TO 128 STEP 8
1320 PLOT 80,g
1330 DRAW 64,0
1340 NEXT g
1350 FOR g=80 TO 144 STEP 8
1360 PLOT g,64
1370 DRAW 0,64
1380 NEXT g
1390 FOR q=1 TO 8
1400 PRINT AT q+5,20;c(q)
1410 NEXT q
1420 PRINT AT 20,8;"CHARACTER #
- ";
1430 PRINT c
1440 PRINT AT 5,5;CHR$ (144+c)
1450 RETURN
1500 REM calculate character val
ues
1510 LET xv=7-(x-10)
1520 LET z=2↑xv
1530 LET v=y-5
1540 IF p=1 THEN GO TO 1600
1550 LET c(v)=c(v)+z
1560 GO TO 1650
1600 REM
1610 LET c(v)=c(v)-z
```

```
1620 REM
1650 FOR q=1 TO 8
1660 PRINT AT q+5,20;"    "
1670 PRINT AT q+5,20;c(q)
1680 NEXT q
1700 REM
1710 LET s=USR CHR$ (144+c)
1720 FOR q=1 TO 8
1730 POKE q+(s-1),c(q)
1740 NEXT q
1750 PRINT AT 5,5;CHR$ (144+c)
1800 RETURN
2000 REM draw border around scre
en
2005 REM
2010 PLOT 0,0
2020 DRAW 255,0
2030 DRAW 0,175
2040 DRAW -255,0
2050 DRAW 0,-175
2060 RETURN
```

BIG CHARACTER

DESCRIPTION

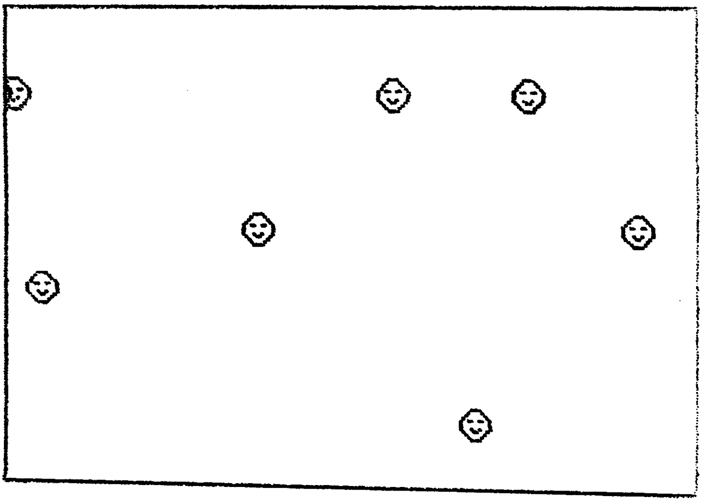
Where your computer scores over many of its rivals is that it has its own built in, high resolution commands without having to add various high resolution packs. The program Big Character displays the use of just one of those commands, namely the PLOT command. This enables us to plot points to the full resolution of the computer, that is 256 pixels by 176 pixels. The routine shown here, in lines 220 to 270, could be used in any program where we require a character that has previously been defined with the use of data statements, to be displayed on the screen at a specified central X, Y coordinate.

RUNNING THE PROGRAM

The data for our large character is stored in the data statements in lines 1000 to 1080. The first two numbers define the size of the character array which we will use to store the data: note that this is dynamically dimensioned on reading that data. Here we are storing the information in binary form: that is, using the digits 0 and 1 to define whether a particular pixel is to be 'lit' or 'unlit'. If you hold the book far enough away from you, you can probably see the character actually drawn out by those data statements. Having stored all the information in the array C(X,Y), we input the variables XC and YC to define the central coordinate for displaying the character, and finally the routine in lines 220 to 270 plots out the character on the screen. Line 300 then sends us back to plot out another one, and so on.

PROGRAM STRUCTURE

60-70	set colours
90	draw border round screen using subroutine at 500
120-180	set up character array from data statements
210	input character centre coordinates
220-270	plot character on screen
300	back again for another go
500-560	border drawing subroutine
1000-1080	data statements for character



```
   1 REM BIG CHARACTER
   2 REM ***********************
   3 REM
  10 REM this program generates
large characters using
  20 REM the plot command, with
character data
  30 REM stored in an array.
  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border
  85 REM
  90 GO SUB 500
  95 REM
 100 REM set up character array
from data statements
 110 REM
 120 READ x,y
 130 DIM c(x,y)
 140 FOR i=1 TO y
 150 FOR j=1 TO x
 160 READ c(j,i)
 170 NEXT j
 180 NEXT i
 190 REM
 200 REM input character coordin
ates and draw character
 205 REM
 210 INPUT xc,yc: REM character
centre cordinates
 220 FOR i=1 TO y
 230 FOR j=1 TO x
 240 IF c(j,i)=0 THEN GO TO 260
 250 PLOT xc-j,yc-i
 260 NEXT j
 270 NEXT i
 300 GO TO 200
 495 REM
 500 REM draw border around scre
en
 505 REM
 510 PLOT 0,0
 520 DRAW 255,0
 530 DRAW 0,175
 540 DRAW -255,0
 550 DRAW 0,-175
 560 RETURN
 995 REM
1000 DATA 12,12
1002 DATA 0,0,0,0,1,1,1,1,0,0,0,
0
1005 DATA 0,0,0,1,0,0,0,0,1,0,0,
0
1007 DATA 0,0,1,0,0,0,0,0,0,1,0,
0
1008 DATA 0,1,0,0,0,0,0,0,0,0,1,
0
```

```
1010 DATA 1,0,0,1,1,0,0,1,1,0,0,
1
1020 DATA 1,0,0,0,0,0,0,0,0,0,0,
1
1030 DATA 1,0,0,0,0,0,0,0,0,0,0,
1
1040 DATA 1,0,0,0,1,0,0,1,0,0,0,
1
1050 DATA 0,1,0,0,0,1,1,0,0,0,1,
0
1060 DATA 0,0,1,0,0,0,0,0,0,1,0,
0
1070 DATA 0,0,0,1,0,0,0,0,1,0,0,
0
1080 DATA 0,0,0,0,1,1,1,1,0,0,0,
0
```

MOVING CHARACTERS

DESCRIPTION

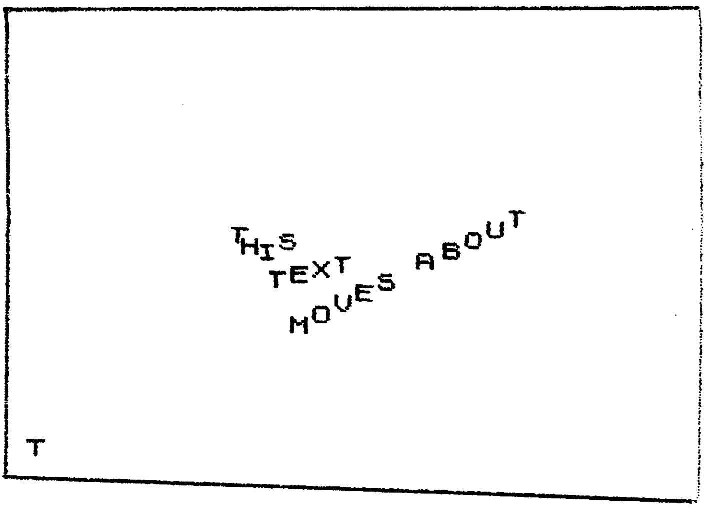
When displaying text on the screen, most people will think conventionally and assume that all text has to be displayed in straight lines, with all the letters being shown like the letters in this book. That is, we simply move across and do not bother putting characters sideways, upside down, or whatever. Normally this is of no great importance, but there are occasions when it would be desirable, and even necessary, to have letters displayed underneath each other, diagonally sloping upwards, or indeed any way we wish. Take the plotting of graphs, when we would like to label the axes properly, perhaps following the slope of a curve for instance. This program, incorporating routines from some of our earlier high resolution programs, does just that.

RUNNING THE PROGRAM

Initially we input the variables X and Y to define the starting point for our character, which is input in line 130 as the variable C$. The movement increment, S, is defined in line 2, and the character C$ stored as an 8 x 8 array C(J,I) using the computer POINT command, in lines 155 to 180. The keyboard is then used to detect which way you would like the character to be moved. This should be familiar to you from earlier programs: pressing 5 to move left, 6 to move down etc. More familiar routines from our high resolution cursor programs follow, to erase the previous character position and restore the screen, and to save the screen contents and plot the character in the new position. When you're happy with the characters position, pressing N will allow you to input a new one.

PROGRAM STRUCTURE

60-70	set colours
90	draw border round screen using subroutine at 800
110-180	input data and set up arrays
210-270	input character movement from keyboard
310-340	check character is within border
410-480	erase previous character position and restore screen
510-550	save screen contents at new character position
610-680	plot character at new position
700	go back for another go
800-860	border drawing subroutine


THIS
TEXT
MOVES ABOUT
T

```
   1 REM MOVING CHARACTERS
   2 REM ************************
   3 REM
  10 REM this program will move
any displayed character
  20 REM with a high resolution
increment, using the
  30 REM keyboard to control the
 movement
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 800
  95 REM
 100 REM input data and set up a
rrays
 105 REM
 110 DIM c(8,8): DIM m(8,8)
 115 REM
 120 INPUT x,y
 130 INPUT c$
 140 PRINT AT 20,1;c$
 150 LET s=2
 155 LET xf=8*x: LET yf=8*(21-y)
 160 FOR i=1 TO 8
 165 FOR j=1 TO 8
 170 LET c(j,i)=POINT (j+8,i+8)
 175 NEXT j
 180 NEXT i
 190 GO TO 500
 195 REM
 200 REM input character movemen
t from keyboard
 205 REM
 210 IF INKEY$="" THEN GO TO 210
 220 LET xo=xf: LET yo=yf
 230 IF INKEY$="5" THEN LET xf=x
f-s
 240 IF INKEY$="6" THEN LET yf=y
f-s
 250 IF INKEY$="7" THEN LET yf=y
f+s
 260 IF INKEY$="8" THEN LET xf=x
f+s
 270 IF INKEY$="n" THEN GO TO 12
0
 295 REM
 300 REM check character is with
in bounds
 305 REM
 310 IF xf<8 THEN LET xf=8
 320 IF xf>247 THEN LET xf=247
 330 IF yf<8 THEN LET yf=8
```

```
340 IF yf>167 THEN LET yf=167
395 REM
400 REM erase previous characte
r position and restore screen
405 REM
410 FOR i=1 TO 8
420 FOR j=1 TO 8
430 IF m(j,i)=0 THEN GO TO 460
440 PLOT j+xo,i+yo
450 GO TO 470
460 PLOT INVERSE 1;j+xo,i+yo
470 NEXT j
480 NEXT i
495 REM
500 REM save screen contents at
new character position
505 REM
510 FOR i=1 TO 8
520 FOR j=1 TO 8
530 LET m(j,i)=POINT (j+xf,i+yf
)
540 NEXT j
550 NEXT i
595 REM
600 REM plot character at new p
osition
605 REM
610 FOR i=1 TO 8
620 FOR j=1 TO 8
630 IF c(j,i)=0 THEN GO TO 660
640 PLOT j+xf,i+yf
650 GO TO 670
660 PLOT INVERSE 1;j+xf,i+yf
670 NEXT j
680 NEXT i
700 GO TO 200
800 REM draw border around scre
en
805 REM
810 PLOT 0,0
820 DRAW 255,0
830 DRAW 0,175
840 DRAW -255,0
850 DRAW 0,-175
860 RETURN
```

SCALING AND STRETCHING

SCALE 1

DESCRIPTION

The ability to scale a shape is one of the most useful in the computer's repertoire, and finds a home in many a program. For instance, Computer Aided Design would not be where it is today without this function. Unfortunately, your computer does not have a scaling command, and hence this routine. In its most simple form as we present it here, scaling just involves taking an object (here we have a rather simplistic view of a tree!), increasing the size of each line that makes up the object, and plotting out our new drawing. What this particular program suffers from is movement of the object as new ones are plotted: in other words, our original design does not get surrounded by larger ones, or itself surrounds smaller ones, but just becomes part of a grand row of small, medium and large trees.

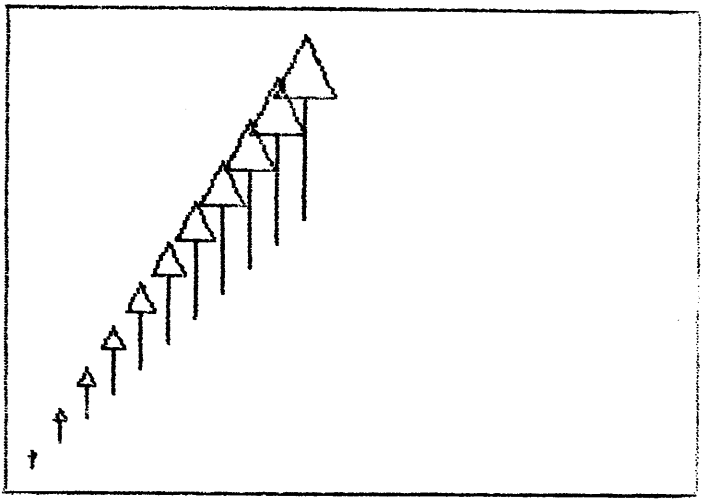
RUNNING THE PROGRAM

In line 110 we dimension our shape data arrays to contain 20 variables each. The data comes from the statements in lines 210 to 250, and as you can see the first number read is the number of sets of data statements to come: in our case 4. Dimensioning to 20 is just a precaution! In order, the data statements present the coordinates X, Y of the start of one of the lines that make up the tree, and the coordinates of the end of that line. Hence, four statements for our four line drawing. The scaling factor S is then input in line 280: when S = 1 we have the original size, a number less than 1 is smaller, and a number greater than 1 gives us a larger image. Scaling factors are then calculated in lines 310 to 360, and our new image plotted out in lines 410 to 530, by drawing out each line in turn. Our usual variable DS is used for dot spacing, and you can specify this to be whatever you like. As pointed our earlier, this program suffers from not having a constant central coordinate.

PROGRAM STRUCTURE

60-70	set colours
90	draw border using subroutine at 800
110-120	set up shape and scaled shape data arrays
140-170	read data for shape
210-250	data for shape
280	input scaling factor
310-360	calculate scaling

410-530	plot each line in turn to specified size
600	go back for another go with a new scaling factor
800-860	border drawing subroutine



```
   1 REM SCALE 1
   2 REM ***********************
   3 REM
  10 REM routine to change the s
cale of a shape in
  20 REM the shape data table
  30 REM
  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 800
  95 REM
 100 REM set up and input data f
or scaling
 105 REM
 106 REM shape data arrays
 107 REM
 110 DIM x(20): DIM y(20): DIM u
(20): DIM v(20)
 115 REM
 116 REM scaled shape data array
s
 117 REM
 120 DIM a(20): DIM b(20): DIM c
(20): DIM d(20)
 125 REM
 130 REM set up shape data array
 135 REM
 140 READ nl: REM number of line
s in shape
 150 FOR i=1 TO nl
 160 READ x(i),y(i),u(i),v(i)
 170 NEXT i
 200 REM shape data
 205 REM
 210 DATA 4
 220 DATA 100,90,100,130
 230 DATA 100,150,90,130
 240 DATA 90,130,110,130
 250 DATA 110,130,100,150
 260 REM
 280 INPUT s: REM scaling factor
 290 REM
 300 REM do scaling
 305 REM
 310 FOR c=1 TO nl
 320 LET a(c)=x(c)*s
 330 LET b(c)=y(c)*s
 340 LET c(c)=u(c)*s
 350 LET d(c)=v(c)*s
 360 NEXT c
 395 REM
 400 REM draw shape
 405 REM
```

```
 410 FOR c=1 TO nl
 420 LET ds=1
 430 LET p=c(c)-a(c)
 440 LET q=d(c)-b(c)
 450 LET r=SQR (p*p+q*q)
 460 LET lx=p/r
 470 LET ly=q/r
 480 FOR i=0 TO r STEP ds
 490 LET x=a(c)+i*lx
 500 LET y=b(c)+i*ly
 510 PLOT x,y
 520 NEXT i
 530 NEXT c
 600 GO TO 280: REM do again
 795 REM
 800 REM draw border around scre
en
 805 REM
 810 PLOT 0,0
 820 DRAW 255,0
 830 DRAW 0,175
 840 DRAW -255,0
 850 DRAW 0,-175
 860 RETURN
```

SCALE 2

DESCRIPTION

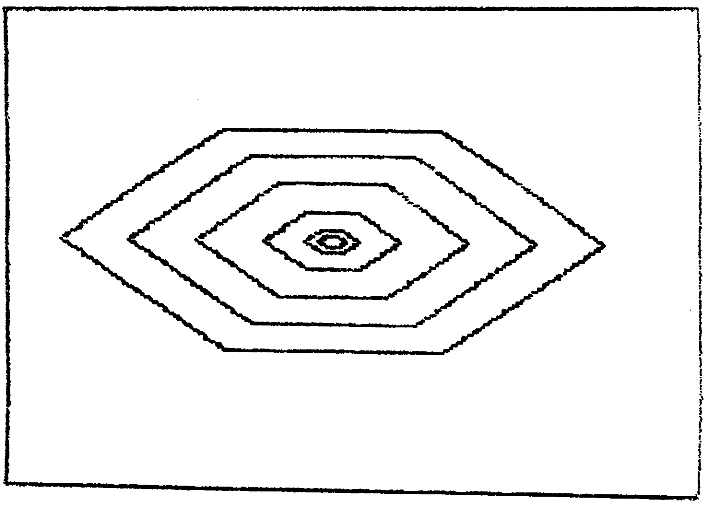
Again here we are taking a shape, and scaling it in both X and Y directions, but with the major fault of the previous program rectified. This time we have a routine to correct the movement of the object as it is scaled, and plot everything out from a common, constant X,Y coordinate. Thus we have the same shape, expanded in both X and Y, or indeed contracted in X and Y, all centred on the same coordinates. This new routine is quite a straightforward 7 line one (lines 310 to 350). One other difference is that our object is this time rather more exotic, being made up of six lines rather than just 4. You can of course experiment with objects that are far more complicated than this: just be careful about the data statements in lines 210 to 255, and make sure you have all the X,Y coordinates right, and more importantly in the right order.

RUNNING THE PROGRAM

As with Scale 1, we dimension our shape and scaled shape data arrays (lines 110 to 120), read in the shape data (lines 140 to 170), and give the data statements (lines 210 to 255). The scaling factor S is input in line 280: as before a number greater than 1 means a larger shape, and less than 1 means a smaller one. The illustration shown ranges from S = 3 down to S = 0.1. The same routines as previously used are here to perform the scaling and draw the shape. The only new one is contained in lines 310 to 350, which calculates the central coordinates for our larger (or smaller) object: these are the variables CX and CY.

PROGRAM STRUCTURE

60-70	set colours
90	draw border using subroutine at 800
110-120	set up shape and scaled shape data arrays
140-170	read data for shape
210-255	data for shape
280	input scaling factor
310-350	calculate new central coordinates
410-460	calculate scaling
510-630	plot each line in turn to specified size
700	go back for another go with a new scaling factor
800-860	border drawing subroutine



```
   1 REM SCALE 2
   2 REM ************************
   3 REM
  10 REM routine to change the s
cale of a shape in
  20 REM the shape data table
  30 REM using the shapes centre

  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 800
  95 REM
 100 REM set up and input data f
or scaling
 105 REM
 106 REM shape data arrays
 107 REM
 110 DIM x(20): DIM y(20): DIM u
(20): DIM v(20)
 115 REM
 116 REM scaled shape data array
s
 117 REM
 120 DIM a(20): DIM b(20): DIM c
(20): DIM d(20)
 125 REM
 130 REM set up shape data array
 135 REM
 140 READ nl: REM number of line
s in shape
 150 FOR i=1 TO nl
 160 READ x(i),y(i),u(i),v(i)
 170 NEXT i
 200 REM shape data
 205 REM
 210 DATA 6
 220 DATA 100,110,140,110
 230 DATA 140,110,170,90
 235 DATA 170,90,140,70
 240 DATA 140,70,100,70
 250 DATA 100,70,70,90
 255 DATA 70,90,100,110
 260 REM
 280 INPUT s: REM scaling factor
 290 REM
 300 REM find centre
 305 REM
 310 LET cx=0: LET cy=0
 320 FOR c=1 TO nl
 330 LET cx=cx+x(c)+u(c)
 335 LET cy=cy+y(c)+v(c)
 340 NEXT c
 345 LET cx=cx/(2*nl)
```

```
350 LET cy=cy/(2*nl)
395 REM
400 REM do scaling
405 REM
410 FOR c=1 TO nl
420 LET a(c)=cx+(cx-x(c))*s
430 LET b(c)=cy+(cy-y(c))*s
440 LET c(c)=cx+(cx-u(c))*s
450 LET d(c)=cy+(cy-v(c))*s
460 NEXT c
495 REM
500 REM draw shape
505 REM
510 FOR c=1 TO nl
520 LET ds=1
530 LET p=c(c)-a(c)
540 LET q=d(c)-b(c)
550 LET r=SQR (p*p+q*q)
560 LET lx=p/r
570 LET ly=q/r
580 FOR i=0 TO r STEP ds
590 LET x=a(c)+i*lx
600 LET y=b(c)+i*ly
610 PLOT x,y
620 NEXT i
630 NEXT c
700 GO TO 280: REM do again
795 REM
800 REM draw border around scre
en
805 REM
810 PLOT 0,0
820 DRAW 255,0
830 DRAW 0,175
840 DRAW -255,0
850 DRAW 0,-175
860 RETURN
```

STRETCH 1

DESCRIPTION

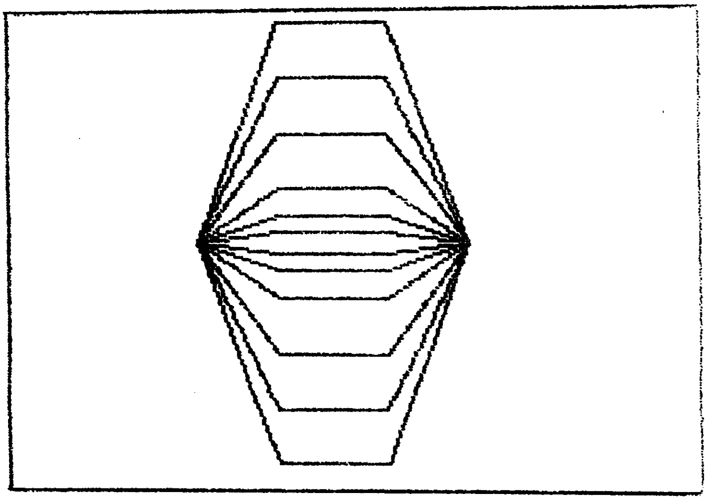
Stretching, although on the surface the same thing as scaling, is in fact a very different animal. Scaling merely produces a larger or smaller image of our original object, based either around the same or a different central coordinate. Stretching, on the other hand, does not necessarily change every line of our object to the same extent, but ideally we do want to stick to the same central coordinates. You can see in the illustrations here that we have a normal image, one stretched in the X axis, and one stretched in the Y axis. With the program being written the way that it has, you can combine stretching in both X and Y axes, without having to use the same stretching factor for each one.

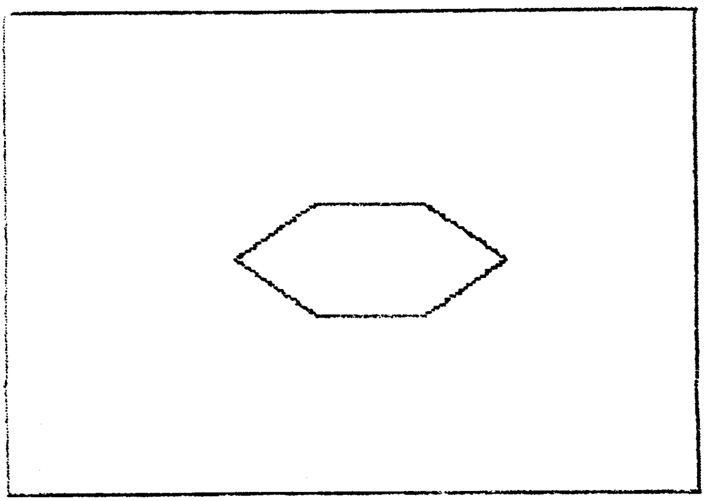
RUNNING THE PROGRAM

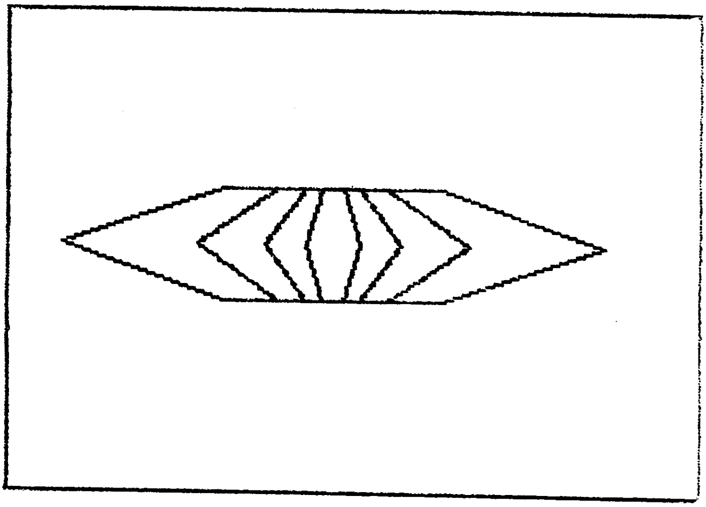
Until we reach line 280 the program follows the same lines as our earlier Scale 2 program. That is, we set up our shape and scaled shape data arrays (lines 110 to 120), and read in the data in lines 210 to 255 by the routine in lines 140 and 170. You will note that we are using the same object as last time, that is, a six sided figure. Line 280 lets us input the scaling factors SX and SY in the X and Y axes, and these are later used in lines 410 and 460 to calculate the scaling and stretching figures. Before and after that we find the central coordinates of our object (lines 310 to 350), and actually plot the figure out (lines 510 to 630) one line at a time.

PROGRAM STRUCTURE

60-70	set colours
90	draw border using subroutine at 800
110-120	set up shape and scaled shape data arrays
140-170	read data for shape
210-255	data for shape
280	input scaling factors
310-350	calculate new central coordinates
410-460	calculate scaling
510-630	plot each line in turn to specified size
700	go back for another go with a new scaling factor
800-860	border drawing subroutine







```
   1 REM STRETCH 1
   2 REM ***********************
   3 REM
  10 REM routine to stretch or c
hange the scale of a shape in
  20 REM the shape data table. I
t uses the
  30 REM shapes centre and diffe
rential X,Y scaling factors.
  40 REM
  50 REM set colours
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 800
  95 REM
 100 REM set up and input data f
or scaling
 105 REM
 106 REM shape data arrays
 107 REM
 110 DIM x(20): DIM y(20): DIM u
(20): DIM v(20)
 115 REM
 116 REM scaled shape data array
s
 117 REM
 120 DIM a(20): DIM b(20): DIM c
(20): DIM d(20)
 125 REM
 130 REM set up shape data array
 135 REM
 140 READ nl: REM number of line
s in shape
 150 FOR i=1 TO nl
 160 READ x(i),y(i),u(i),v(i)
 170 NEXT i
 200 REM shape data
 205 REM
 210 DATA 6
 220 DATA 100,110,140,110
 230 DATA 140,110,170,90
 235 DATA 170,90,140,70
 240 DATA 140,70,100,70
 250 DATA 100,70,70,90
 255 DATA 70,90,100,110
 260 REM
 280 INPUT sx,sy: REM scaling fa
ctors in X and Y axis
 290 REM
 300 REM find centre
 305 REM
 310 LET cx=0: LET cy=0
 320 FOR c=1 TO nl
 330 LET cx=cx+x(c)+u(c)
 335 LET cy=cy+y(c)+v(c)
```

```
 340 NEXT c
 345 LET cx=cx/(2*nl)
 350 LET cy=cy/(2*nl)
 395 REM
 400 REM do scaling and stretchi
ng
 405 REM
 410 FOR c=1 TO nl
 420 LET a(c)=cx+(cx-x(c))*sx
 430 LET b(c)=cy+(cy-y(c))*sy
 440 LET c(c)=cx+(cx-u(c))*sx
 450 LET d(c)=cy+(cy-v(c))*sy
 460 NEXT c
 495 REM
 500 REM draw shape
 505 REM
 510 FOR c=1 TO nl
 520 LET ds=1
 530 LET p=c(c)-a(c)
 540 LET q=d(c)-b(c)
 550 LET r=SQR (p*p+q*q)
 560 LET lx=p/r
 570 LET ly=q/r
 580 FOR i=0 TO r STEP ds
 590 LET x=a(c)+i*lx
 600 LET y=b(c)+i*ly
 610 PLOT x,y
 620 NEXT i
 630 NEXT c
 700 GO TO 280: REM do again
 795 REM
 800 REM draw border around scre
en
 805 REM
 810 PLOT 0,0
 820 DRAW 255,0
 830 DRAW 0,175
 840 DRAW -255,0
 850 DRAW 0,-175
 860 RETURN
```

STRETCH 2

DESCRIPTION

The Stretch 1 program as described is an extremely useful one, but alas it is not without its limitations. Although we can stretch images in both X and Y directions, one thing which we do not have control over is the angle of stretching. At present, everything is going at ninety degree angles. What if, as is very common in Computer Aided Design, and indeed other fields, we want to stretch something at, say, 37 degrees to the X axis? The routine in lines 410 to 650 in this program performs just that function. I will not go into the mathematical detail here, many excellent books have been written on the subject, but will simply say that it works!

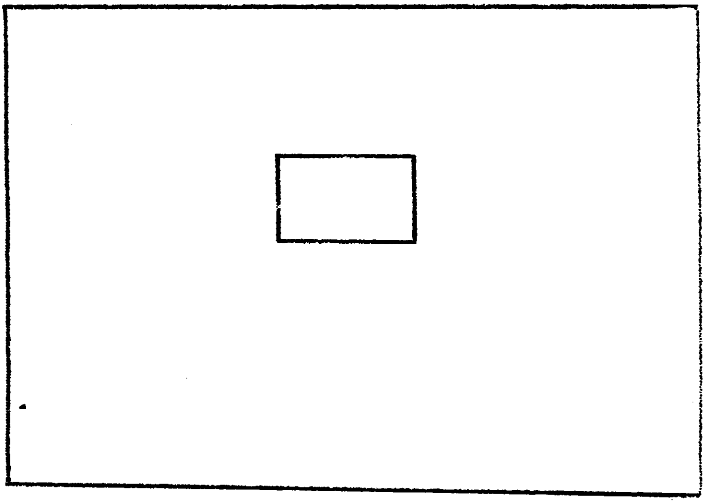
RUNNING THE PROGRAM

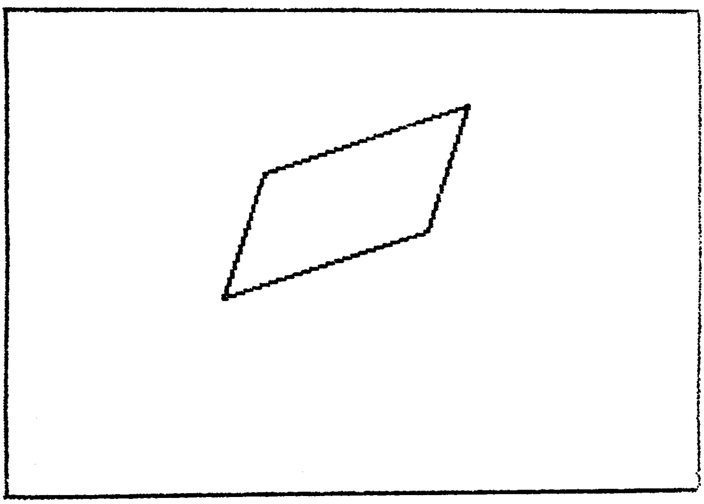
As in previous programs, we first of all set up the shape and scaled shape data arrays before reading in the actual data itself from lines 210 to 240. This time we revert to a much simpler shape, that of a rectangle. In line 280 we again input the scaling factors in the X and Y axes, and in line 290 we input AS, the angle of stretching. This is the angle by which we will evaluate our shape above the X axis. In other words, if AS is equal to 45 degrees, as it is in the illustration, the line joining the two corners of the rectangle will be at 45 degrees to the X axis. After calculating the centre of the newly formed shape, the scaling, stretching and rotating routine in lines 410 to 650 comes into effect. As you can see this is quite complicated, and I do not intend to go into any detail. This book is designed to help you with graphics on the computer, not to give a thesis on mathematical theory!

PROGRAM STRUCTURE

60-70	set colours
90	draw border using routine at 1000
110-120	dimension shape and scaled shape data arrays
140-170	read shape data
210-240	shape data statements
280	input scaling factors in X and Y axis
285	input angle of rotation
286	convert degrees to radians
310-350	calculate centre coordinates
410-650	perform scaling, stretching and rotation calculations
710-830	draw new shape line by line

900	go back for another go
1000-1060	border drawing subroutine





```
1 REM STRETCH 2
2 REM ************************
3 REM
10 REM routine to stretch or c
hange the scale of a shape in
20 REM the shape data table. I
t uses the
30 REM shapes centre and diffe
rential X,Y scaling factors
40 REM plus an angle of rotati
on along which stretching takes
45 REM place.
46 REM
50 REM set colours
60 INK 0
70 PAPER 7
75 REM
80 REM draw border around scre
en
85 REM
90 GO SUB 1000
95 REM
100 REM set up and input data f
or scaling
105 REM
106 REM shape data arrays
107 REM
110 DIM x(20): DIM y(20): DIM u
(20): DIM v(20)
115 REM
116 REM scaled shape data array
s
117 REM
120 DIM a(20): DIM b(20): DIM c
(20): DIM d(20)
125 REM
130 REM set up shape data array
135 REM
140 READ nl: REM number of line
s in shape
150 FOR i=1 TO nl
160 READ x(i),y(i),u(i),v(i)
170 NEXT i
200 REM shape data
205 REM
210 DATA 4
220 DATA 100,120,150,120
230 DATA 150,120,150,90
235 DATA 150,90,100,90
240 DATA 100,90,100,120
260 REM
280 INPUT sx,sy: REM scaling fa
ctors in X and Y axis
285 INPUT as: REM angle of stre
tching
286 LET as=as*3.14159/180
290 REM
300 REM find centre
305 REM
```

```
310 LET cx=0: LET cy=0
320 FOR c=1 TO nl
330 LET cx=cx+x(c)+u(c)
335 LET cy=cy+y(c)+v(c)
340 NEXT c
345 LET cx=cx/(2*nl)
350 LET cy=cy/(2*nl)
395 REM
400 REM do scaling and stretchi
ng
405 REM
410 FOR i=1 TO nl
420 LET x1=x(i)-cx
430 LET y1=y(i)-cy
440 LET f=(x1*COS (as)+y1*SIN (
as))*sy
450 LET g=(-x1*SIN (as)+y1*COS
(as))*SX
460 LET x2=f*COS (as)-g*SIN (as
)
470 LET a(i)=x2+cx
480 LET y2=f*SIN (as)+g*COS (as
)
490 LET b(i)=y2+cy
500 LET x1=u(i)-cx
510 LET y1=v(i)-cy
520 LET f=(x1*COS (as)+y1*SIN
as))*sy
530 LET g=(-x1*SIN (as)+y1*COS
(as))*sx
540 LET x2=f*COS (as)-g*SIN (as
)
550 LET c(i)=x2+cx
560 LET y2=f*SIN (as)+g*COS (as
)
570 LET d(i)=y2+cy
650 NEXT i
700 REM draw shape
705 REM
710 FOR c=1 TO nl
720 LET ds=1
730 LET p=c(c)-a(c)
740 LET q=d(c)-b(c)
750 LET r=SQR (p*p+q*q)
760 LET lx=p/r
770 LET ly=q/r
780 FOR i=0 TO r STEP ds
790 LET x=a(c)+i*lx
800 LET y=b(c)+i*ly
810 PLOT x,y
820 NEXT i
830 NEXT c
900 GO TO 280: REM do again
995 REM
1000 REM draw border around scre
en
1005 REM
1010 PLOT 0,0
```

```
1020 DRAW 255,0
1030 DRAW 0,175
1040 DRAW -255,0
1050 DRAW 0,-175
1060 RETURN
```

ROTATING AND MOVING

ROTATE

DESCRIPTION

In this section we introduce the concept of a transformation matrix. A transformation matrix is essentially a set of equations which are applied to a coordinate point in order to move it to the required position. I shall not endeavour to derive these equations (there are many excellent books on the subject), but simply show how they can be used to produce the required effects. The rotational transformation matrix consists of four equations and these are calculated in lines 250 to 280. Lines 290-300 use the values from this matrix to calculate the new coordinates of the point.

Rotation requires the movement of a point in a circle around a fixed axis on the screen. By making the point the end coordinate of a line, a line or a shape can be rotated around this axis. The axis of rotation can lie anywhere on the screen, it may even lie on the same coordinates as the point to be rotated. In this program you will notice that the small cross is being rotated in a clockwise direction around an axis thereby describing a circle, note that the point erase — lines 310 to 317 — were removed to produce the diagram. Counterclockwise rotation can be produced by using a negative angle of rotation.

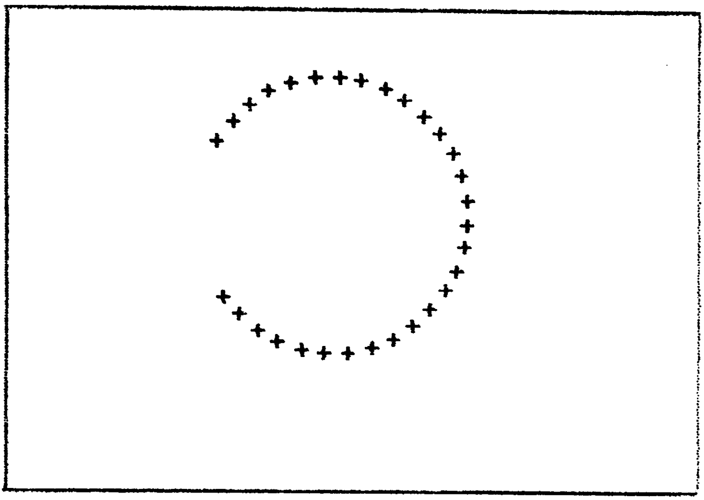
RUNNING THE PROGRAM

The program requires the input of five parameters. These five are the X and Y coordinates of the centre of rotation, the X and Y coordinates of the point to be rotated and the angle of rotation. The angle of rotation is in degrees and is the angle between two lines drawn from the centre of rotation to the 0 degree or three o'clock position and from the centre to the new point position. It should be noted that the FOR NEXT loop in lines 235 and 410 are inserted to generate a sequence of 360 rotational plot points, these should be removed to plot a single rotation.

PROGRAM STRUCTURE

60-70	set colours
90	draw border around screen using subroutine at 500
110	input coordinates for centre of rotation
120	input coordinates for point to be plotted
130	input angle of rotation
150	set up array for rotation matrix

210	convert rotation angle from degrees to radians
215-220	initialise variables
225	plot point at centre of rotation
230	set start angle at 0
235	loop to plot 365 consecutive rotations
240	add angle of rotation to start angle
250-280	calculate rotational transform matrix
290-300	calculate new coordinate point position
310-317	erase previous rotated point position
330-400	plot new rotated point
410	loop to rotate again by the rotation angle
500-560	border drawing subroutine



```
   1 REM ROTATE
   2 REM ***********************
   3 REM
  10 REM this program rotates a
point around
  20 REM a central point on the
screen
  30 REM
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 500
  95 REM
 100 REM input parameters
 110 INPUT xc,yc: REM coordinate
s of centre of rotation
 120 INPUT xp,yp: REM coordinate
s of point to be rotated
 130 INPUT ar: REM angle of rota
tion
 150 DIM m(2,2)
 195 REM
 200 REM rotate point
 205 REM
 210 LET ar=ar*3.14159/180
 215 LET xr=xp: LET yr=yp
 217 LET xo=xr: LET yo=yr
 220 LET xp=-(xc-xp): LET yp=-(y
c-yp)
 225 PLOT xc,yc
 230 LET r=0
 235 FOR q=1 TO 360
 240 LET r=r+ar
 250 LET m(1,1)=COS (r)
 260 LET m(1,2)=SIN (r)
 270 LET m(2,1)=-SIN (r)
 280 LET m(2,2)=COS (r)
 290 LET x=xc+xp*m(1,1)+yp*m(2,1
)
 300 LET y=yc+xp*m(1,2)+yp*m(2,2
)
 310 PLOT INVERSE 1;xo-2,yo
 313 DRAW INVERSE 1;4,0
 315 PLOT INVERSE 1;xo,yo-2
 317 DRAW INVERSE 1;0,4
 330 PLOT xr-2,yr
 333 DRAW 4,0
 335 PLOT xr,yr-2
 337 DRAW 0,4
 350 LET xo=xr: LET yo=yr
 400 LET xr=x: LET yr=y
 410 NEXT q
 420 GO TO 100
```

```
 500 REM draw border around scre
en
 510 PLOT 0,0
 520 DRAW 255,0
 530 DRAW 0,175
 540 DRAW -255,0
 550 DRAW 0,-175
 560 RETURN
```

ROTATE 2

DESCRIPTION

In the same way that the program ROTATE rotated a point around a fixed axis on the screen we can also rotate a line about a fixed axis. This is not difficult since one is simply rotating two points — the two end coordinates of the line. It should be noted that in this program the line start and end coordinates are both input as relative coordinates. A relative coordinate means that the coordinate is not the normal screen coordinate but a value which is relative to the coordinate of the axis point. If the axis is set at the absolute screen coordinates of X = 100 and Y = 80 then to have the start of the line at the absolute screen coordinates of X = 150 and Y = 100 gives us a relative coordinate value of X = 50 and Y = 20. From this we can see that the relative coordinates are obtained by this calculation:

coordinate of point — axis coordinate

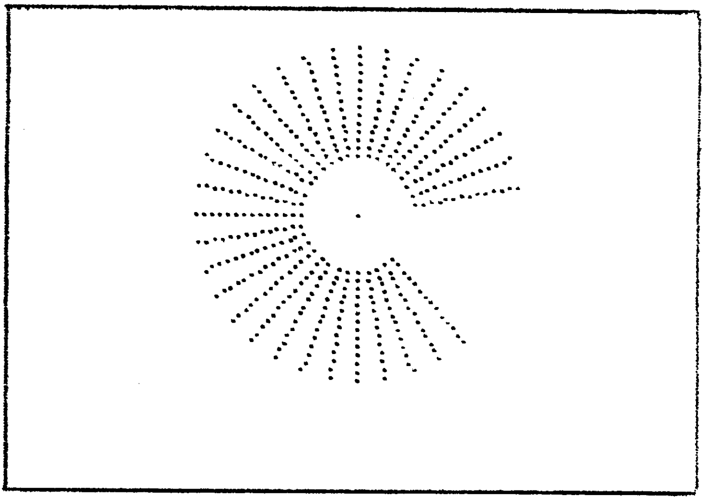
RUNNING THE PROGRAM

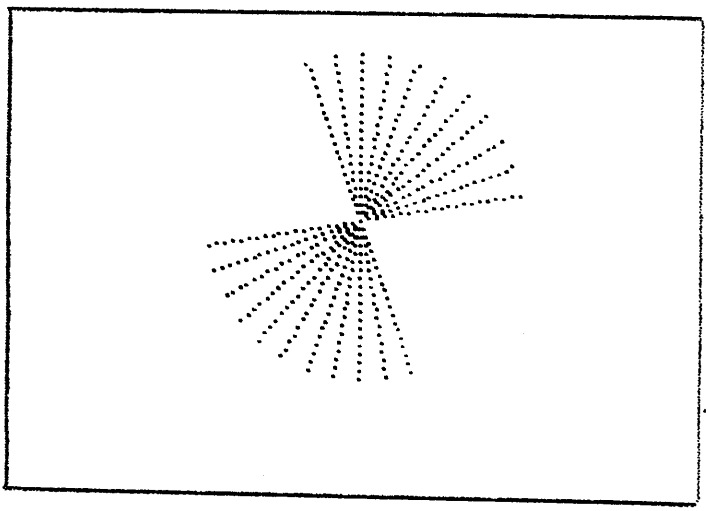
The program requires the input of seven parameters, they are as follows. The X and Y coordinates of the central axis around which the line is rotated. This is followed by the X and Y coordinates of the start of the line and then the X and Y coordinates of the end of the line, all four values being relative coordinates with respect to the centre of rotation. The last parameter value is the angle of rotation, this is in degrees and is the angle between two lines drawn from the centre of rotation to the original dot position and from the centre to the new dot position. Note that the FOR NEXT loop in lines 235 and 500 have been inserted to generate a sequence of fifty rotations of the increment angle. These should be removed to plot a single rotation.

PROGRAM STRUCTURE

60-70	set colours
90	draw border around screen using subroutine at 700
110	input coordinates for centre of rotation
120	input relative coordinates for start of line
125	input relative coordinates for end of line
130	input angle of rotation
150	set up array for rotation matrix
210	convert angle to radians

215	initialise variables
225	plot point at centre of rotation
230	set start angle at zero
235	loop to plot 50 consecutive rotation increments
240	add angle of rotation to start angle
250-280	calculate rotational transform matrix
290-340	calculate new coordinate point positions
360-460	routine to draw line between the two end points
500	loop to next rotation increment
700-760	border drawing subroutine





```
1 REM ROTATE 2
2 REM ***********************
3 REM
10 REM this program rotates a line around
20 REM a central point on the screen
30 REM
40 REM
50 REM set colours
55 REM
60 INK 0
70 PAPER 7
75 REM
80 REM draw border around screen
85 REM
90 GO SUB 700
95 REM
100 REM input parameters
110 INPUT xc,yc: REM coordinates of centre of rotation
120 INPUT xp,yp: REM relative line start coordinates
125 INPUT xq,yq: REM relative line end coordinates
130 INPUT ar: REM angle of rotation
150 DIM m(2,2)
195 REM
200 REM rotate line
205 REM
210 LET ar=ar*3.14159/180
215 LET xr=xp: LET yr=yp
225 PLOT xc,yc
230 LET r=0
235 FOR z=1 TO 50
240 LET r=r+ar
250 LET m(1,1)=COS (r)
260 LET m(1,2)=SIN (r)
270 LET m(2,1)=-SIN (r)
280 LET m(2,2)=COS (r)
290 LET x=xc+xp*m(1,1)+yp*m(2,1)
300 LET y=yc+xp*m(1,2)+yp*m(2,2)
310 LET xb=x: LET yb=y
320 LET x=xc+xq*m(1,1)+yq*m(2,1)
330 LET y=yc+xq*m(1,2)+yq*m(2,2)
340 LET xe=x: LET ye=y
345 REM
350 REM draw line
355 REM
360 LET ds=3
370 LET p=xe-xb
380 LET q=ye-yb
390 LET rl=SQR (p*p+q*q)
```

```
400 LET lx=p/rl
410 LET ly=q/rl
420 FOR i=0 TO rl STEP ds
430 LET x=xb+i*lx
440 LET y=yb+i*ly
450 PLOT x,y
460 NEXT i
500 NEXT z
695 REM
700 REM draw border around scre
en
705 REM
710 PLOT 0,0
720 DRAW 255,0
730 DRAW 0,175
740 DRAW -255,0
750 DRAW 0,-175
760 RETURN
```

ROTATE 3

DESCRIPTION

In the same way that the program ROTATE 2 rotated a line around a fixed axis on the screen we can also rotate a shape about a fixed axis. This is not difficult since one is simply rotating a set of lines, each line being specified by the two end coordinates of the line. The data for the shape is stored in a shape table, this is stored in one of three arrays. The other two arrays are used to store the data for the rotated shape and the previous rotation – this is required by the routine which erases the previous rotation. The data is stored as the beginning X and Y coordinate of a line followed by the end X and Y coordinates of the same line, these four values are then repeated for each line in the shape. In this program the shape data is obtained from a set of data statements — lines 710 to 740. The set of displays which accompany this program show how by varying the centre of rotation the shape is rotated in different ways, depending on whether the rotational centre lies within the shape, directly on a line of axis through the shape or to one side of the shape, also shown is that the lines used to draw the shape can have a variable dot spacing.

RUNNING THE PROGRAM

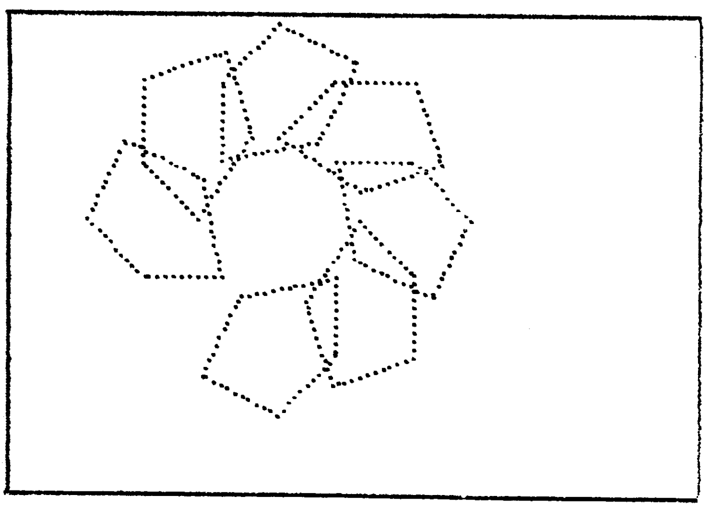
All the parameters required by the program are stored directly within the program. The X and Y coordinates of the central axis around which the shape is rotated is stored as the variables xc and yc in line 255. The number of lines in the shape is stored as variable nl in line 240. The X and Y coordinates of the start and end of each line are stored as data statements in lines 710 to 740. The last parameter value is the angle of rotation, this is in degrees and is stored as the variable ar in line 296.

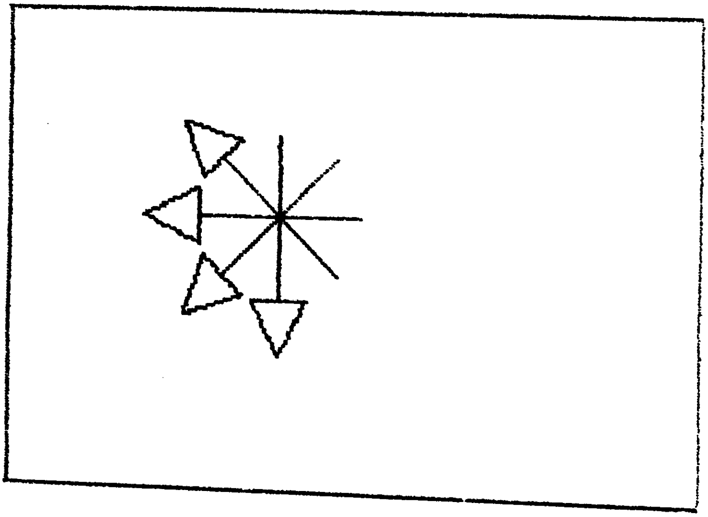
Note: that the FOR NEXT loop in lines 300 and 620 have been inserted to generate a sequence of fifty rotations of the increment angle. These should be removed to plot a single rotation. When plotting shapes with more than 20 lines then the size of the shape data arrays should be increased accordingly.

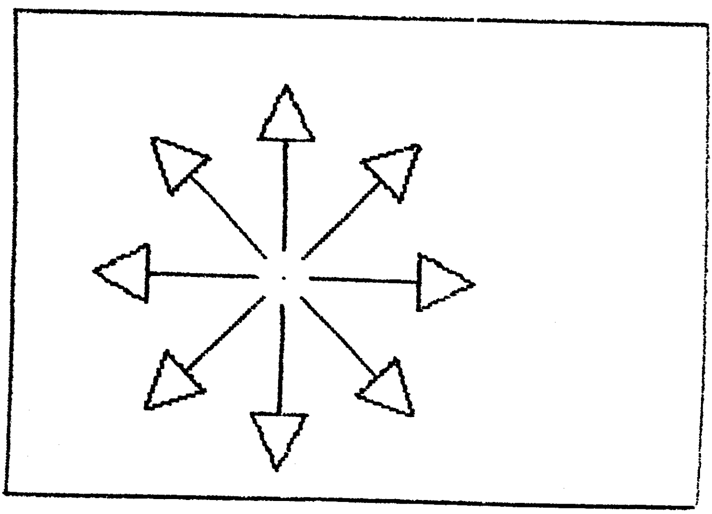
PROGRAM STRUCTURE

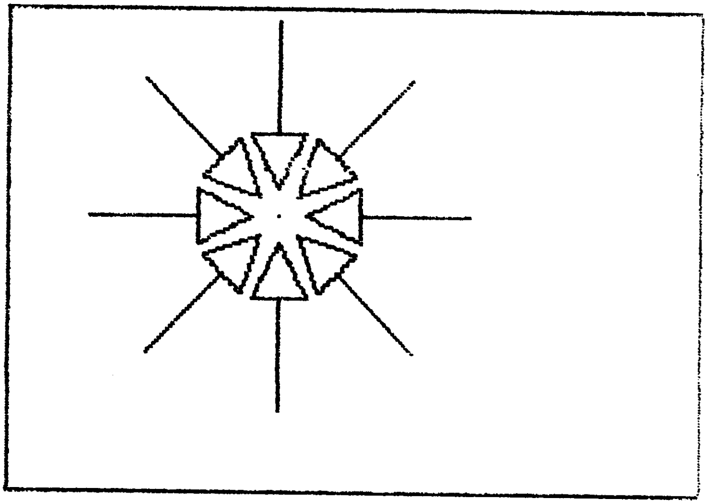
60-70	set colours
90	draw border around screen using subroutine at 900
110	set up array for rotation matrix
120-150	matrix for original data shape

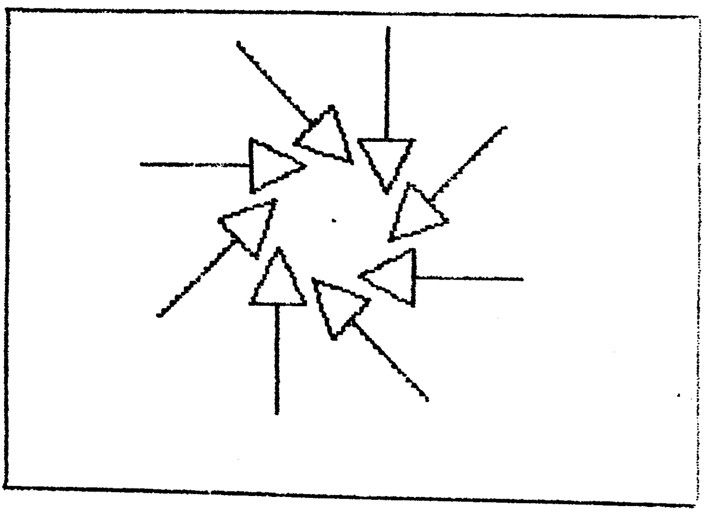
160-190	matrix for erased shape data
210-225	matrix for displayed shape data
234-296	initialise variables and constants
240	number of lines in shape
255	plot point at centre of rotation
260-290	load coordinate data into original shape matrix
296	set start angle to zero
300	loop to plot 50 consecutive rotation increments
310	add angle of rotation to start angle
320-350	calculate rotational transform matrix
370-500	calculate new coordinate point positions
520	jump to routine to draw lines
560-610	put displayed shape data into erased shape matrix
620	loop to next rotation increment
710-740	shape table data
900-960	border drawing subroutine
1000-1140	subroutine to draw shape
2000-2140	subroutine to erase shape











```
   1 REM ROTATE 3
   2 REM ***********************
   3 REM
  10 REM program to rotate a 2 D
 object about
  20 REM a point on the screen.
  30 REM
  40 REM
  50 REM set colours
  55 REM
  60 INK 0
  70 PAPER 7
  75 REM
  80 REM draw border around scre
en
  85 REM
  90 GO SUB 900
  95 REM
 100 REM arrays for data transfo
rmation
 105 REM
 106 REM rotation matrix
 107 REM
 110 DIM m(2,2)
 115 REM
 116 REM original shape data
 117 REM
 120 DIM x(20)
 130 DIM y(20)
 135 REM
 140 DIM u(20)
 150 DIM v(20)
 155 REM
 156 REM erased shape data
 157 REM
 160 DIM w(20)
 170 DIM z(20)
 175 REM
 180 DIM s(20)
 190 DIM t(20)
 200 REM
 205 REM displayed shape data
 206 REM
 210 DIM o(20)
 215 DIM p(20)
 220 DIM q(20)
 225 DIM r(20)
 230 REM
 234 REM set up constants and da
ta from data tables
 235 REM
 240 LET nl=4
 250 LET xc=100: LET yc=80
 255 PLOT xc,yc
 260 FOR n=1 TO nl
 270 READ x(n),y(n),u(n),v(n)
 280 LET w(n)=x(n): LET z(n)=y(n
): LET s(n)=u(n): LET t(n)=v(n)
 290 NEXT n
```

```
296 LET ar=45: LET r=0
297 REM
298 LET ar=ar*3.14159/180
299 REM
300 FOR a=1 TO 50
305 REM
310 LET r=r+ar
315 REM
316 REM set up rotation matrix
317 REM
320 LET m(1,1)=COS (r)
330 LET m(1,2)=SIN (r)
340 LET m(2,1)=-SIN (r)
350 LET m(2,2)=COS (r)
360 REM
365 REM rotate shape ar degrees
366 REM
370 FOR n=1 TO nl
380 LET p=-(xc-x(n))
390 LET q=-(yc-y(n))
400 LET x=xc+p*m(1,1)+q*m(2,1)
410 LET y=yc+p*m(1,2)+q*m(2,2)
420 LET o(n)=x
430 LET p(n)=y
440 LET p=-(xc-u(n))
450 LET q=-(yc-v(n))
460 LET x=xc+p*m(1,1)+q*m(2,1)
470 LET y=yc+p*m(1,2)+q*m(2,2)
480 LET q(n)=x
490 LET r(n)=y
500 NEXT n
510 REM
520 GO SUB 2000
530 REM
540 GO SUB 1000
550 REM
560 FOR n=1 TO nl
570 LET w(n)=o(n)
580 LET z(n)=p(n)
590 LET s(n)=q(n)
600 LET t(n)=r(n)
610 NEXT n
620 NEXT a
630 STOP
695 REM
700 REM shape data
705 REM
710 DATA 100,90,100,130
720 DATA 100,150,90,130
730 DATA 90,130,110,130
740 DATA 110,130,100,150
895 REM
900 REM draw border around scre
en
905 REM
910 PLOT 0,0
920 DRAW 255,0
930 DRAW 0,175
940 DRAW -255,0
```

```
 950 DRAW 0,-175
 960 RETURN
 995 REM
1000 REM draw shape
1005 REM
1010 FOR n=1 TO nl
1020 LET ds=1
1030 LET p=q(n)-o(n)
1040 LET q=r(n)-p(n)
1050 LET rl=SQR (p*p+q*q)
1060 LET lx=p/rl
1070 LET ly=q/rl
1080 FOR i=0 TO rl STEP ds
1090 LET x=o(n)+i*lx
1100 LET y=p(n)+i*ly
1110 PLOT x,y
1120 NEXT i
1130 NEXT n
1140 RETURN
1995 REM
2000 REM erase shape
2005 REM
2010 FOR n=1 TO nl
2020 LET ds=1
2030 LET p=s(n)-w(n)
2040 LET q=t(n)-z(n)
2050 LET rl=SQR (p*p+q*q)
2060 LET lx=p/rl
2070 LET ly=q/rl
2080 FOR i=0 TO rl STEP ds
2090 LET x=w(n)+i*lx
2100 LET y=z(n)+i*ly
2110 PLOT INVERSE 1;x,y
2115 LET xc=100: LET yc=100
2120 NEXT i
2130 NEXT n
2140 RETURN
```

MOVE

DESCRIPTION

The application of the transformation matrix can be expanded to cover all manipulation of a shape, not just rotation but also movement (known as translation) and scaling. The primary purpose of this program is to show how a shape can be moved about the screen, but it also imbodies the capability of scaling and rotation. The transformation matrix consists of six quotations. These equations are stored in lines 3000 to 3100. Notice that equations 1 to 4 consist of the rotational transform equation multiplied by a scaling factor, equations 5 and 6 do the movement by adding an offset to the shape position. The program can display any two dimensional shapes. This shape can be moved to any part of the screen, rotated through 360 degrees and stretched in either X or Y axis or both.

RUNNING THE PROGRAM

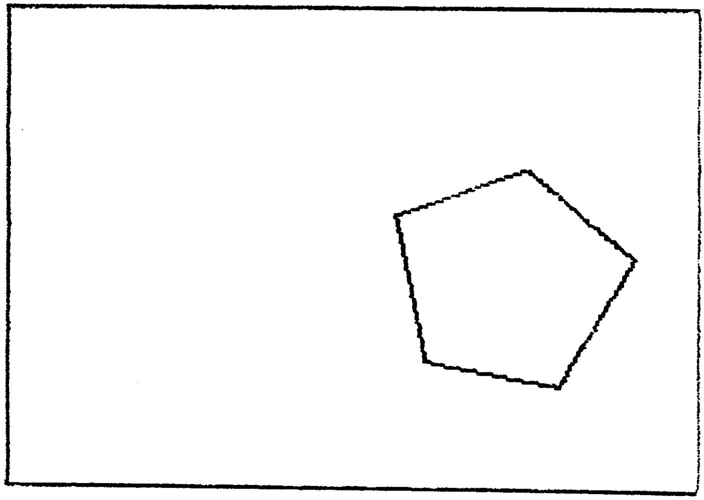
There are no input parameter values since they are all within the program as LET statements. There are six parameter values which control the movement, rotation or scaling of the shape, these are set in lines 120 to 160. Lines 120 and 130 contain the X and Y scaling factors — full size = 1, half size = .5 etc. The rotational angle of the shape is stored as the variable rz in line 140, note that since this angle must be in radians it is multiplied by 3.14159/180. The movement of the shape in the X and Y axis is stored in lines 150 and 160, and is the number of pixels in either direction from the original coordinates stored in the shape table.

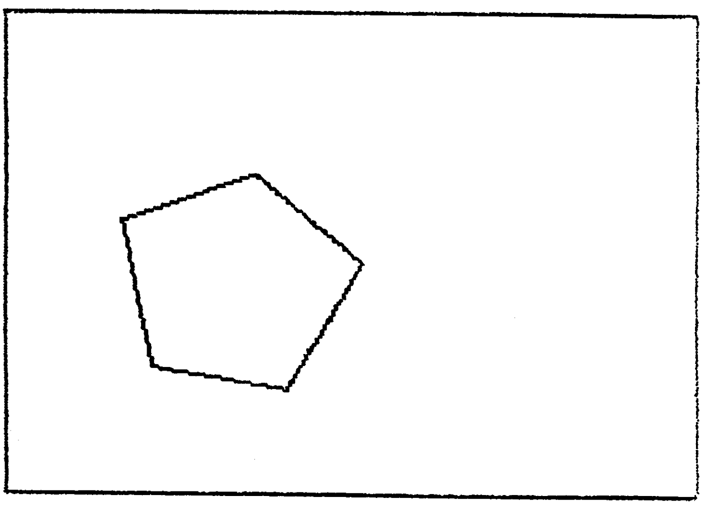
The object shape is stored in a shape table. This table consists simply of the X and Y coordinates of the end of each line comprising the shape. It should be noted that there are one more pair of coordinates than there are lines in the shape, the number of lines in the shape is stored as the variable np as the first value in the data table. The data table is stored as data statements in lines 1110 to 1130. Try designing your own shapes using graph paper and then entering the new values into the data statements.

PROGRAM STRUCTURE

90	draw border around screen using subroutine at 400
110	set up transform matrix array
120 - 130	X and Y scaling factors

140	angle of shape rotation in radians
150 - 160	X and Y axis movement of shape from initial position
210 - 260	main program execution loop
400 - 460	border drawing subroutine
1000-1050	load shape data into arrays — arrays X and Y contain the original shape data — arrays U and V contain the transformed shape data
1110-1130	data statements containing shape data — line 110 contains the number of lines in the shape
2000-2080	find the centre of the shape
3000-3100	perform transformation matrix calculations
4000-4070	performs the trasformation on each coordinate point point within the shape table
5000-5220	draws the shape using the transformed data in the arrays U and V, note lines 5120 and 5130 check that the shape does not fall outside the screen area





```
1 REM MOVE
2 REM ************************
3 REM
10 REM this program uses matri
x transformation to
20 REM move, rotate, or scale
a two dimensional shape
30 REM
40 REM
50 REM
80 REM draw border
90 GO SUB 400
95 REM
100 REM set up constants variab
les and arrays
105 REM
110 DIM a(3,3)
120 LET sx=1
130 LET sy=1
140 LET rz=80*3.14159/180
150 LET tx=-50
160 LET ty=2
190 REM
200 REM main program loop
205 REM
210 GO SUB 1000
220 GO SUB 2000
230 GO SUB 3000
240 GO SUB 4000
250 GO SUB 5000
260 STOP
395 REM
400 REM border drawing subrouti
ne
405 REM
410 PLOT 0,0
420 DRAW 255,0
430 DRAW 0,175
440 DRAW -255,0
450 DRAW 0,-175
460 RETURN
1000 REM initialise shape
1005 REM
1010 READ np
1020 DIM x(np+1): DIM y(np+1): D
IM u(np+1): DIM v(np+1)
1030 FOR i=1 TO np+1
1040 READ x(i),y(i)
1050 NEXT i
1090 REM
1100 REM shape data
1105 REM
1110 DATA 5
1120 DATA 100,100,150,120,175,75
1130 DATA 150,30,100,50,100,100
1200 RETURN
1995 REM
2000 REM find centre of shape
2005 REM
2010 LET cx=0: LET cy=0
```

```
2020 FOR c=1 TO np
2030 LET cx=cx+x(c)
2040 LET cy=cy+y(c)
2050 NEXT c
2060 LET cx=cx/np
2070 LET cy=cy/np
2080 RETURN
2995 REM
3000 REM set transformation matr
ix
3005 REM
3010 LET a(1,1)=sx*COS (rz)
3020 LET a(1,2)=sx*SIN (rz)
3030 REM
3040 LET a(2,1)=sy*(-SIN (rz))
3050 LET a(2,2)=sy*COS (rz)
3060 REM
3070 LET a(3,1)=tx
3080 LET a(3,2)=ty
3090 REM
3100 RETURN
3995 REM
4000 REM do transformation
4005 REM
4010 FOR q=1 TO np+1
4020 LET xt=x(q)-cx
4030 LET yt=y(q)-cy
4040 LET u(q)=cx+(xt*a(1,1)+yt*a
(2,1)+a(3,1))
4050 LET v(q)=cy+(xt*a(1,2)+yt*a
(2,2)+a(3,2))
4060 NEXT q
4070 RETURN
4995 REM
5000 REM draw shape
5005 REM
5010 FOR q=1 TO np
5020 LET xb=u(q): LET yb=v(q)
5030 LET xe=u(q+1): LET ye=v(q+1
)
5040 LET p=xe-xb
5050 LET o=ye-yb
5060 LET r=SQR (p*p+o*o)
5070 LET lx=p/r
5080 LET ly=o/r
5090 FOR i=0 TO r STEP 1
5100 LET x=xb+i*lx
5110 LET y=yb+i*ly
5120 IF x>255 THEN LET x=255
5130 IF y>175 THEN LET y=175
5190 PLOT x,y
5200 NEXT i
5210 NEXT q
5220 RETURN
```

3D DISPLAYS

THREE DIMENSIONAL SHAPE 1

DESCRIPTION

The application of the transformation matrix can be expanded further to cover the generation of three dimensional shapes — it should be noted that they are displayed two dimensionally but optically appear to represent three dimensional objects. To do this simply requires the addition of an extra axis — the Z axis — to the X and Y axis used in a two dimensional transformation matrix. The transformation matrix consists of sixteen equations, they are stored in lines 3000 to 3190. I shall not attempt to explain the mathematics, for those interested I would suggest one of the text books on the subject — 'Principles of Interactive Graphics' by Newman and Sproul.

RUNNING THE PROGRAM

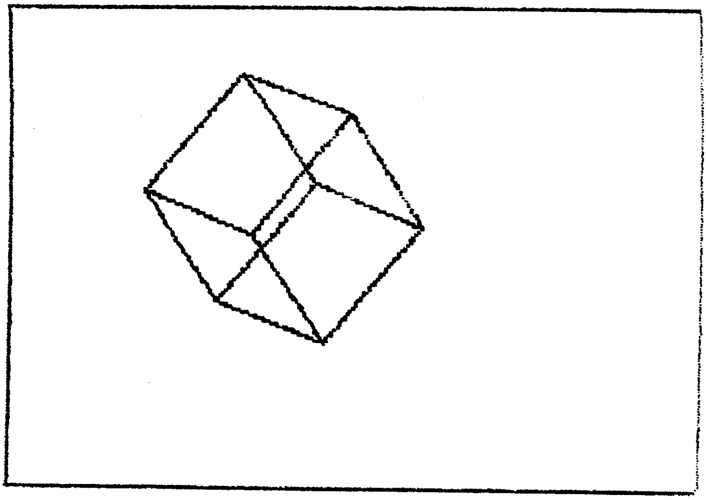
There are no input parameter values since they are all within the program as LET statements. There are nine parameter values which control the movement, rotation or scaling of the shape, these are set in lines 120 to 200. Lines 120 and 140 contain the X,Y and Z scaling factors — full size = 1, half size = .5 etc. The rotational angle of the shape in either one of the three axis are stored in lines 180 to 200, note that since these angles must be in radians they are multiplied by 3.14159/180. The movement of the shape in the X,Y and Z axis is stored in lines 150 to 170, and is the number of pixels in either direction from the original coordinates stored in the shape table.

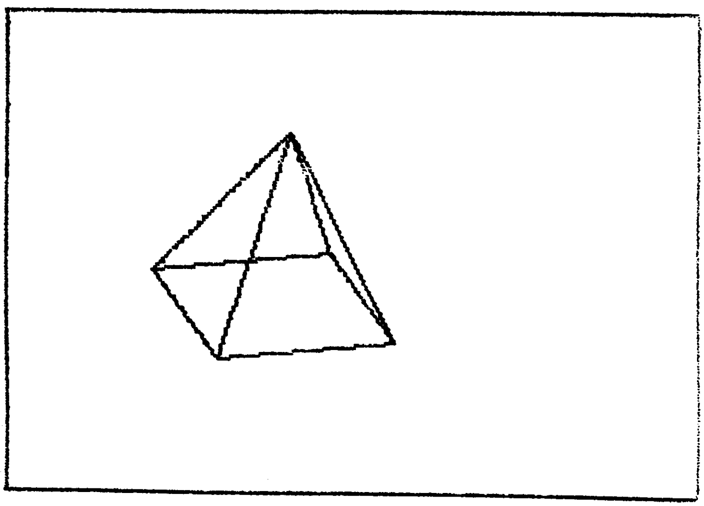
The object shape is stored in a shape table. This table consists of two parts the first is simply of the X,Y and Z coordinates, of each corner coordinate comprising the shape. The second part is a table of connections of pairs of points between which a line should be drawn. The number of edges in the shape is stored as the variable 'ne' and the number of coordinate points between which the edges are connected is stored as variable 'np'. The coordinate table is stored as data statements in lines 1210 to 1220, and the connection table in lines 1310 to 1330.

PROGRAM STRUCTURE

70	draw border around screen using subroutine at 900
100-110	set up transform matrix arrays
120-140	X,Y and Z scaling factors

150-170	X,Y and Z axis movement of shape from initial position
180-200	angle of X,Y and Z axis rotation in radians
410-450	main program execution loop
900-960	border drawing subroutine
1000-1050	load shape data into arrays — arrays S contains the coordinate table of the original shape — array E contains the line connection data — array M contains the transformed coordinate data
1200-1220	data statements containing coordinate shape data as X,Y and Z for each corner point, note that the first three values comprise the coordinates for point 1, the cond three for point 2 etc
2000-2240	draw the shape
3000-3160	perform transformation matrix calculations
3200-3350	set up scaling and translation matrix
4000-4080	performs the transformation on each coordinate point within the shape table
5000-5090	find centre of shape





```
   1 REM 3D DRAWING 1
   2 REM ************************
   3 REM
  10 REM a three dimensional sha
pe is drawn by this program
  20 REM the rotation position a
nd scale of the object
  30 REM can be changed to give
different viewing angles.
  40 REM
  50 REM
  60 REM draw border around scre
en
  65 REM
  70 GO SUB 900
  80 REM
  90 REM set up constants variab
les and arrays
  95 REM
 100 DIM a(4,4)
 110 DIM b(4,4)
 120 LET sx=.3
 130 LET sy=.3
 140 LET sz=.3
 150 LET tx=1
 160 LET ty=1
 170 LET tz=1
 180 LET rx=40*3.14159/180
 190 LET ry=20*3.14159/180
 200 LET rz=50*3.14159/180
 400 REM main program loop
 410 GO SUB 1000
 420 GO SUB 5000
 430 GO SUB 3000
 440 GO SUB 4000
 450 GO SUB 2000
 500 STOP
 895 REM
 900 REM border drawing subrouti
ne
 905 REM
 910 PLOT 0,0
 920 DRAW 255,0
 930 DRAW 0,175
 940 DRAW -255,0
 950 DRAW 0,-175
 960 RETURN
 995 REM
1000 REM initialise shape
1005 REM
1010 LET np=8
1020 LET ne=12
1030 REM
1040 DIM s(3,np)
1050 DIM e(ne,2)
1060 DIM m(3,np)
1100 REM
1110 FOR n=1 TO np
1120 READ s(1,n),s(2,n),s(3,n)
```

```
1130 NEXT n
1150 FOR e=1 TO ne
1160 READ e(e,1),e(e,2)
1170 NEXT e
1195 REM
1200 REM x,y,z point coordinates
1205 REM
1210 DATA 0,0,200,200,0,200,200,
0,0,0,0,0
1220 DATA 0,200,200,200,200,200,
200,200,0,0,200,0
1295 REM
1300 REM connection data
1305 REM
1310 DATA 1,2,2,3,3,4,4,1
1320 DATA 5,1,2,6,4,8,7,3
1330 DATA 6,5,5,8,8,7,7,6
1900 RETURN
1995 REM
2000 REM draw shape
2005 REM
2020 FOR e=1 TO ne
2030 LET v1=e(e,1)
2040 LET v2=e(e,2)
2045 IF v1=0 THEN GO TO 2240
2050 LET xb=m(1,v1)
2060 LET yb=m(2,v1)
2070 LET xe=m(1,v2)
2080 LET ye=m(2,v2)
2090 LET ds=1
2100 LET p=xe-xb
2110 LET q=ye-yb
2120 LET r=SQR (p*p+q*q)
2130 LET lx=p/r
2140 LET ly=q/r
2150 FOR i=0 TO r STEP ds
2160 LET x=xb+i*lx
2170 LET y=yb+i*ly
2180 IF x>255 THEN GO TO 2230
2190 IF y>175 THEN GO TO 2230
2200 IF x<0 THEN GO TO 2230
2210 IF y<0 THEN GO TO 2230
2220 PLOT x,y
2230 NEXT i
2240 NEXT e
2900 RETURN
2995 REM
3000 REM set transformation matr
ix
3005 REM
3010 LET a(1,1)=COS (ry)*COS (rz
)
3020 LET a(1,2)=COS (ry)*SIN (rz
)
3030 LET a(1,3)=-SIN (ry)
3040 LET a(1,4)=0
3050 LET a(2,1)=COS (rx)*(-SIN (
rz))+SIN (rx)*SIN (ry)*COS (rz)
```

```
3060 LET a(2,2)=COS (rx)*COS (rz
)+SIN (rx)*SIN (ry)*SIN (rz)
3070 LET a(2,3)=SIN (rx)*COS (ry
)
3080 LET a(2,4)=0
3090 LET a(3,1)=(-SIN (rx))*(-SI
N (rz))+COS (rx)*SIN (ry)*COS (r
z)
3100 LET a(3,2)=-SIN (rx)*COS (r
z)+COS (rz)*SIN (ry)*SIN (rz)
3110 LET a(3,3)=COS (rx)*COS (ry
)
3120 LET a(3,4)=0
3130 LET a(4,1)=0
3140 LET a(4,2)=0
3150 LET a(4,3)=0
3160 LET a(4,4)=1
3195 REM
3200 REM set up scaling and tran
slation matrix
3205 REM
3210 LET b(1,1)=sx*a(1,1)
3220 LET b(1,2)=sx*a(1,2)
3230 LET b(1,3)=sx*a(1,3)
3240 REM
3250 LET b(2,1)=sy*a(2,1)
3260 LET b(2,2)=sy*a(2,2)
3270 LET b(2,3)=sy*a(2,3)
3280 REM
3290 LET b(3,1)=sz*a(3,1)
3300 LET b(3,2)=sz*a(3,2)
3310 LET b(3,3)=sz*a(3,3)
3320 REM
3330 LET b(4,1)=tx
3340 LET b(4,2)=ty
3350 LET b(4,3)=tz
3900 RETURN
3995 REM
4000 REM perform translation
4005 REM
4010 FOR q=1 TO np
4015 REM
4020 LET xt=s(1,q)-xc
4030 LET yt=s(2,q)-yc
4040 LET zt=s(3,q)-zc
4045 REM
4050 LET m(1,q)=xc+(xt*b(1,1)+yt
*b(2,1)+zt*b(3,1)+b(4,1))
4060 LET m(2,q)=yc+(xt*b(1,2)+yt
*b(2,2)+zt*b(3,2)+b(4,2))
4070 LET m(3,q)=zc+(xt*b(1,3)+yt
*b(2,3)+zt*b(3,3)+b(4,3))
4080 NEXT q
4900 RETURN
4995 REM
5000 REM find centroid
5005 REM
5010 LET p=0: LET q=0: LET r=0
5020 FOR i=1 TO np
```

```
5030 LET p=p+s(1,i)
5040 LET q=q+s(2,i)
5050 LET r=r+s(3,i)
5060 NEXT i
5070 LET xc=p/np
5080 LET yc=q/np
5090 LET zc=r/np
5900 RETURN
```

THREE DIMENSIONAL SHAPE 2

DESCRIPTION

This program is identical to the program THREE DIMENSIONAL SHAPE 1 except that an additional subroutine has been added to remove hidden lines. Hidden lines are those lines which lie out of sight of the viewer and are hidden behind the front surfaces. By removing these hidden lines the shape of the object becomes much clearer. The subroutine which checks for hidden lines is located between line numbers 6000 and 6140.

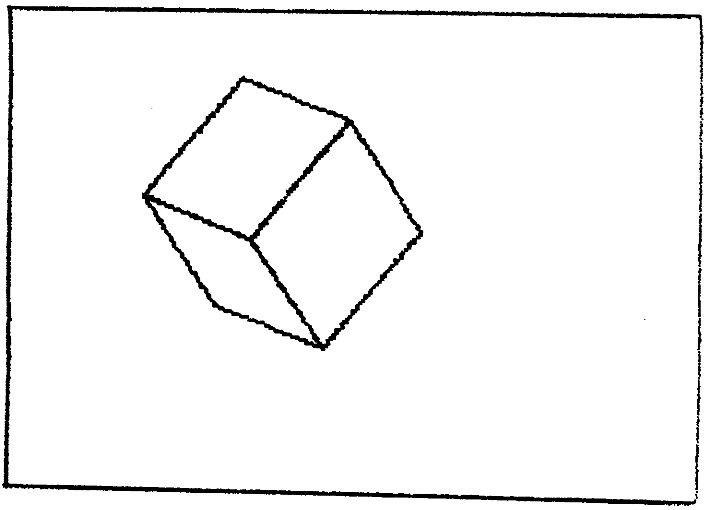
RUNNING THE PROGRAM

The parameters and data tables required by this program are the same as those used for the program THREE DIMENSIONAL SHAPE 1, consult this program for information. Note that the connection table now describes object faces rather than lines.

PROGRAM STRUCTURE

Lines 1 to 5995 are identical to THREE DIMENSIONAL SHAPE 1 (consult for details).

6000-6140 subroutine to check for hidden surfaces



```
   1 REM 3D DRAWING 2
   2 REM ************************
   3 REM
  10 REM a three dimensional sha
pe is drawn by this program
  20 REM the rotation position a
nd scale of the object
  30 REM can be changed to give
different viewing angles.
  40 REM the program incorporate
s a routine to remove hidden lin
es
  50 REM
  60 REM draw border around scre
en
  65 REM
  70 GO SUB 900
  80 REM
  90 REM set up constants variab
les and arrays
  95 REM
 100 DIM a(4,4)
 110 DIM b(4,4)
 115 DIM c(3)
 117 DIM d(3)
 120 LET sx=.3
 130 LET sy=.3
 140 LET sz=.3
 150 LET tx=1
 160 LET ty=1
 170 LET tz=1
 180 LET rx=40*3.14159/180
 190 LET ry=20*3.14159/180
 200 LET rz=50*3.14159/180
 400 REM main program loop
 410 GO SUB 1000
 420 GO SUB 5000
 430 GO SUB 3000
 440 GO SUB 4000
 450 GO SUB 6000
 500 STOP
 895 REM
 900 REM border drawing subrouti
ne
 905 REM
 910 PLOT 0,0
 920 DRAW 255,0
 930 DRAW 0,175
 940 DRAW -255,0
 950 DRAW 0,-175
 960 RETURN
 995 REM
1000 REM initialise shape
1005 REM
1010 LET np=8
1020 LET ne=4
1030 LET nf=6
1040 DIM s(3,np)
1050 DIM e(nf,ne,2)
```

```
1060 DIM m(3,np)
1100 REM
1110 FOR n=1 TO np
1120 READ s(1,n),s(2,n),s(3,n)
1130 NEXT n
1140 FOR f=1 TO nf
1150 FOR e=1 TO ne
1160 READ e(f,e,1),e(f,e,2)
1170 NEXT e
1180 NEXT f
1195 REM
1200 REM x,y,z point coordinates
1205 REM
1210 DATA 0,0,200,200,0,200,200,
0,0,0,0,0
1220 DATA 0,200,200,200,200,200,
200,200,0,0,200,0
1295 REM
1300 REM connection data
1305 REM
1310 DATA 1,2,2,3,3,4,4,1
1320 DATA 5,1,1,4,4,8,8,5
1330 DATA 6,5,5,8,8,7,7,6
1340 DATA 2,6,6,7,7,3,3,2
1350 DATA 1,5,5,6,6,2,2,1
1360 DATA 3,7,7,8,8,4,4,3
1900 RETURN
1995 REM
2000 REM draw shape
2005 REM
2020 FOR e=1 TO ne
2030 LET v1=e(f,e,1)
2040 LET v2=e(f,e,2)
2045 IF v1=0 THEN GO TO 2240
2050 LET xb=m(1,v1)
2060 LET yb=m(2,v1)
2070 LET xe=m(1,v2)
2080 LET ye=m(2,v2)
2090 LET ds=1
2100 LET p=xe-xb
2110 LET q=ye-yb
2120 LET r=SQR (p*p+q*q)
2130 LET lx=p/r
2140 LET ly=q/r
2150 FOR i=0 TO r STEP ds
2160 LET x=xb+i*lx
2170 LET y=yb+i*ly
2180 IF x>255 THEN GO TO 2230
2190 IF y>175 THEN GO TO 2230
2200 IF x<0 THEN GO TO 2230
2210 IF y<0 THEN GO TO 2230
2220 PLOT x,y
2230 NEXT i
2240 NEXT e
2900 RETURN
2995 REM
3000 REM set transformation matr
ix
3005 REM
```

```
3010 LET a(1,1)=COS (ry)*COS (rz
)
3020 LET a(1,2)=COS (ry)*SIN (rz
)
3030 LET a(1,3)=-SIN (ry)
3040 LET a(1,4)=0
3050 LET a(2,1)=COS (rx)*(-SIN (
rz))+SIN (rx)*SIN (ry)*COS (rz)
3060 LET a(2,2)=COS (rx)*COS (rz
)+SIN (rx)*SIN (ry)*SIN (rz)
3070 LET a(2,3)=SIN (rx)*COS (ry
)
3080 LET a(2,4)=0
3090 LET a(3,1)=(-SIN (rx))*(-SI
N (rz))+COS (rx)*SIN (ry)*COS (r
z)
3100 LET a(3,2)=-SIN (rx)*COS (r
z)+COS (rz)*SIN (ry)*SIN (rz)
3110 LET a(3,3)=COS (rx)*COS (ry
)
3120 LET a(3,4)=0
3130 LET a(4,1)=0
3140 LET a(4,2)=0
3150 LET a(4,3)=0
3160 LET a(4,4)=1
3195 REM
3200 REM set up scaling and tran
slation matrix
3205 REM
3210 LET b(1,1)=sx*a(1,1)
3220 LET b(1,2)=sx*a(1,2)
3230 LET b(1,3)=sx*a(1,3)
3240 REM
3250 LET b(2,1)=sy*a(2,1)
3260 LET b(2,2)=sy*a(2,2)
3270 LET b(2,3)=sy*a(2,3)
3280 REM
3290 LET b(3,1)=sz*a(3,1)
3300 LET b(3,2)=sz*a(3,2)
3310 LET b(3,3)=sz*a(3,3)
3320 REM
3330 LET b(4,1)=tx
3340 LET b(4,2)=ty
3350 LET b(4,3)=tz
3900 RETURN
3995 REM
4000 REM perform translation
4005 REM
4010 FOR q=1 TO np
4015 REM
4020 LET xt=s(1,q)-xc
4030 LET yt=s(2,q)-yc
4040 LET zt=s(3,q)-zc
4045 REM
4050 LET m(1,q)=xc+(xt*b(1,1)+yt
*b(2,1)+zt*b(3,1)+b(4,1))
4060 LET m(2,q)=yc+(xt*b(1,2)+yt
*b(2,2)+zt*b(3,2)+b(4,2))
```

```
4070 LET m(3,q)=zc+(xt*b(1,3)+yt
*b(2,3)+zt*b(3,3)+b(4,3))
4080 NEXT q
4900 RETURN
4995 REM
5000 REM find centroid
5005 REM
5010 LET p=0: LET q=0: LET r=0
5020 FOR i=1 TO np
5030 LET p=p+s(1,i)
5040 LET q=q+s(2,i)
5050 LET r=r+s(3,i)
5060 NEXT i
5070 LET xc=p/np
5080 LET yc=q/np
5090 LET zc=r/np
5900 RETURN
5995 REM
6000 REM hidden surface check
6005 REM
6010 FOR f=1 TO nf
6020 FOR j=1 TO 3
6030 LET c(j)=m(j,e(f,1,2))-m(j,
e(f,1,1))
6040 LET d(j)=m(j,e(f,2,1))-m(j,
e(f,2,2))
6050 NEXT j
6060 LET p1=c(2)*d(3)-c(3)*d(2)
6070 LET p2=c(3)*d(1)-c(1)*d(3)
6080 LET p3=c(1)*d(2)-c(2)*d(1)
6090 LET q1=1-m(1,e(f,1,2))
6100 LET q2=1-m(2,e(f,1,2))
6110 LET q3=500-m(3,e(f,1,2))
6120 LET w=p1*q1+p2*q2+p3*q3
6130 IF w>=0 THEN GO SUB 2000
6140 NEXT f
6900 RETURN
```

THREE DIMENSIONAL SHAPE 3

DESCRIPTION

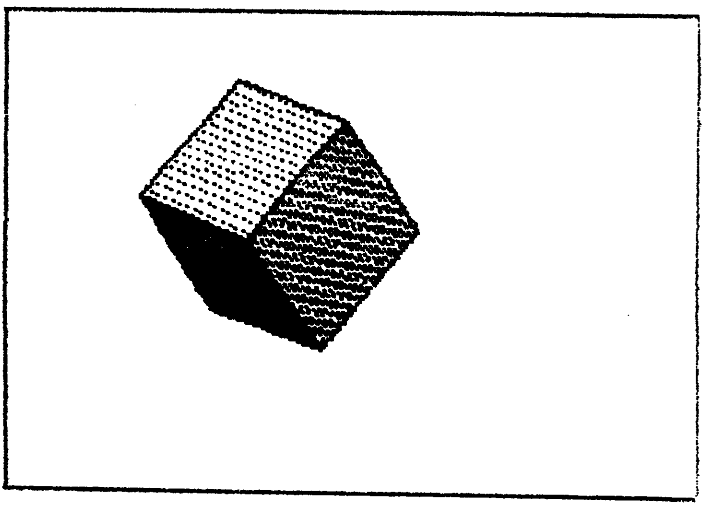
This program is identical to the program THREE DIMENSIONAL SHAPE 1 except that additional subroutines have been added to remove hidden lines, and to shade the faces of the displayed surfaces in respect of incedent light coming from above in the Y axis. By shading the surfaces the viewer becomes fully aware of the shape of the three dimensional object as well as adding realism to the display.

RUNNING THE PROGRAM

The parameters and data tables required by this program are the same as those used for the program THREE DIMENSIONAL SHAPE 2 (consult this program for information).

PROGRAM STRUCTURE

Lines 1 to 5995 are identical to THREE DIMENSIONAL SHAPE 1 (consult for details).
6000-6140 subroutine to check for hidden surfaces
7000-7330 shade the displayed surfaces



```
   1 REM 3D DRAWING 3
   2 REM ***********************
   3 REM
  10 REM a three dimensional sha
pe is drawn by this program
  20 REM the rotation position a
nd scale of the object
  30 REM can be changed to give
different viewing angles.
  40 REM the program incorporate
s a routine to remove hidden lin
es
  50 REM the displayed faces are
shaded in respect of incident li
ght
  52 REM coming from above in th
e Y axis.
  55 REM
  60 REM draw border around scre
en
  65 REM
  70 GO SUB 900
  80 REM
  90 REM set up constants variab
les and arrays
  95 REM
 100 DIM a(4,4)
 110 DIM b(4,4)
 115 DIM c(3)
 117 DIM d(3)
 120 LET sx=.3
 130 LET sy=.3
 140 LET sz=.3
 150 LET tx=1
 160 LET ty=1
 170 LET tz=1
 180 LET rx=40*3.14159/180
 190 LET ry=20*3.14159/180
 200 LET rz=50*3.14159/180
 400 REM main program loop
 410 GO SUB 1000
 420 GO SUB 5000
 430 GO SUB 3000
 440 GO SUB 4000
 450 GO SUB 6000
 500 STOP
 895 REM
 900 REM border drawing subrouti
ne
 905 REM
 910 PLOT 0,0
 920 DRAW 255,0
 930 DRAW 0,175
 940 DRAW -255,0
 950 DRAW 0,-175
 960 RETURN
 995 REM
1000 REM initialise shape
1005 REM
```

```
1010 LET np=8
1020 LET ne=4
1030 LET nf=6
1040 DIM s(3,np)
1050 DIM e(nf,ne,2)
1060 DIM m(3,np)
1100 REM
1110 FOR n=1 TO np
1120 READ s(1,n),s(2,n),s(3,n)
1130 NEXT n
1140 FOR f=1 TO nf
1150 FOR e=1 TO ne
1160 READ e(f,e,1),e(f,e,2)
1170 NEXT e
1180 NEXT f
1195 REM
1200 REM x,y,z point coordinates
1205 REM
1210 DATA 0,0,200,200,0,200,200,
0,0,0,0,0
1220 DATA 0,200,200,200,200,200,
200,200,0,0,200,0
1295 REM
1300 REM connection data
1305 REM
1310 DATA 1,2,2,3,3,4,4,1
1320 DATA 5,1,1,4,4,8,8,5
1330 DATA 6,5,5,8,8,7,7,6
1340 DATA 2,6,6,7,7,3,3,2
1350 DATA 1,5,5,6,6,2,2,1
1360 DATA 3,7,7,8,8,4,4,3
1900 RETURN
1995 REM
2000 REM draw shape
2005 REM
2020 FOR e=1 TO ne
2030 LET v1=e(f,e,1)
2040 LET v2=e(f,e,2)
2045 IF v1=0 THEN GO TO 2240
2050 LET xb=m(1,v1)
2060 LET yb=m(2,v1)
2070 LET xe=m(1,v2)
2080 LET ye=m(2,v2)
2090 LET ds=1
2100 LET p=xe-xb
2110 LET q=ye-yb
2120 LET r=SQR (p*p+q*q)
2130 LET lx=p/r
2140 LET ly=q/r
2150 FOR i=0 TO r STEP ds
2160 LET x=xb+i*lx
2170 LET y=yb+i*ly
2180 IF x>255 THEN GO TO 2230
2190 IF y>175 THEN GO TO 2230
2200 IF x<0 THEN GO TO 2230
2210 IF y<0 THEN GO TO 2230
2220 PLOT x,y
2230 NEXT i
2240 NEXT e
```

```
2300 GO SUB 7000
2900 RETURN
2995 REM
3000 REM set transformation matr
ix
3005 REM
3010 LET a(1,1)=COS (ry)*COS (rz
)
3020 LET a(1,2)=COS (ry)*SIN (rz
)
3030 LET a(1,3)=-SIN (ry)
3040 LET a(1,4)=0
3050 LET a(2,1)=COS (rx)*(-SIN (
rz))+SIN (rx)*SIN (ry)*COS (rz)
3060 LET a(2,2)=COS (rx)*COS (rz
)+SIN (rx)*SIN (ry)*SIN (rz)
3070 LET a(2,3)=SIN (rx)*COS (ry
)
3080 LET a(2,4)=0
3090 LET a(3,1)=(-SIN (rx))*(-SI
N (rz))+COS (rx)*SIN (ry)*COS (r
z)
3100 LET a(3,2)=-SIN (rx)*COS (r
z)+COS (rz)*SIN (ry)*SIN (rz)
3110 LET a(3,3)=COS (rx)*COS (ry
)
3120 LET a(3,4)=0
3130 LET a(4,1)=0
3140 LET a(4,2)=0
3150 LET a(4,3)=0
3160 LET a(4,4)=1
3195 REM
3200 REM set up scaling and tran
slation matrix
3205 REM
3210 LET b(1,1)=sx*a(1,1)
3220 LET b(1,2)=sx*a(1,2)
3230 LET b(1,3)=sx*a(1,3)
3240 REM
3250 LET b(2,1)=sy*a(2,1)
3260 LET b(2,2)=sy*a(2,2)
3270 LET b(2,3)=sy*a(2,3)
3280 REM
3290 LET b(3,1)=sz*a(3,1)
3300 LET b(3,2)=sz*a(3,2)
3310 LET b(3,3)=sz*a(3,3)
3320 REM
3330 LET b(4,1)=tx
3340 LET b(4,2)=ty
3350 LET b(4,3)=tz
3900 RETURN
3995 REM
4000 REM perform translation
4005 REM
4010 FOR q=1 TO np
4015 REM
4020 LET xt=s(1,q)-xc
4030 LET yt=s(2,q)-yc
4040 LET zt=s(3,q)-zc
```

```
4045 REM
4050 LET m(1,q)=xc+(xt*b(1,1)+yt
*b(2,1)+zt*b(3,1)+b(4,1))
4060 LET m(2,q)=yc+(xt*b(1,2)+yt
*b(2,2)+zt*b(3,2)+b(4,2))
4070 LET m(3,q)=zc+(xt*b(1,3)+yt
*b(2,3)+zt*b(3,3)+b(4,3))
4080 NEXT q
4900 RETURN
4995 REM
5000 REM find centroid
5005 REM
5010 LET p=0: LET q=0: LET r=0
5020 FOR i=1 TO np
5030 LET p=p+s(1,i)
5040 LET q=q+s(2,i)
5050 LET r=r+s(3,i)
5060 NEXT i
5070 LET xc=p/np
5080 LET yc=q/np
5090 LET zc=r/np
5900 RETURN
5995 REM
6000 REM hidden surface check
6005 REM
6010 FOR f=1 TO nf
6020 FOR j=1 TO 3
6030 LET c(j)=m(j,e(f,1,2))-m(j,
e(f,1,1))
6040 LET d(j)=m(j,e(f,2,1))-m(j,
e(f,2,2))
6050 NEXT j
6060 LET p1=c(2)*d(3)-c(3)*d(2)
6070 LET p2=c(3)*d(1)-c(1)*d(3)
6080 LET p3=c(1)*d(2)-c(2)*d(1)
6090 LET q1=1-m(1,e(f,1,2))
6100 LET q2=1-m(2,e(f,1,2))
6110 LET q3=500-m(3,e(f,1,2))
6120 LET w=p1*q1+p2*q2+p3*q3
6130 IF w>=0 THEN GO SUB 2000
6140 NEXT f
6900 RETURN
6995 REM
7000 REM shading
7005 REM
7010 LET r1=m(1,e(f,2,1))-m(1,e(
f,1,1))
7020 LET r2=m(2,e(f,2,1))-m(2,e(
f,1,1))
7030 LET r3=m(3,e(f,2,1))-m(3,e(
f,1,1))
7040 LET w1=SQR (r1*r1+r2*r2+r3*
r3)
7050 LET r4=m(1,e(f,4,1))-m(1,e(
f,1,1))
7060 LET r5=m(2,e(f,4,1))-m(2,e(
f,1,1))
7070 LET r6=m(3,e(f,4,1))-m(3,e(
f,1,1))
```

```
7080 LET w2=SQR (r4*r4+r5*r5+r6*
r6)
7090 LET r1=r1/w1
7100 LET r2=r2/w1
7110 LET r3=r3/w1
7120 LET r4=r4/w2
7130 LET r5=r5/w2
7140 LET r6=r6/w2
7150 LET u=r3*r4-r1*r6
7160 IF u<=-.9 THEN RETURN
7170 IF u>-.9 AND u<-.7 THEN LET
 ds=4
7180 IF u>-.7 AND u<-.5 THEN LET
 ds=4
7190 IF u>-.5 AND u<-.3 THEN LET
 ds=3
7200 IF u>-.3 AND u<-.1 THEN LET
 ds=3
7210 IF u>-.1 AND u<+.1 THEN LET
 ds=3
7220 IF u>.1 AND u<.3 THEN LET d
s=2
7230 IF u>.3 AND u<.5 THEN LET d
s=2
7240 IF u>.5 AND u<.7 THEN LET d
s=1
7250 IF u>.7 AND u<.9 THEN LET d
s=1
7260 IF u>.9 THEN LET ds=1
7270 FOR i=1 TO w1 STEP ds
7280 FOR q=1 TO w2 STEP ds
7290 LET x=m(1,e(f,1,1))+i*r1+q*
r4
7300 LET y=m(2,e(f,1,1))+i*r2+q*
r5
7310 PLOT x,y
7320 NEXT q
7330 NEXT i
7900 RETURN
```

THREE DIMENSIONAL SHAPE 4

DESCRIPTION

Perspective is that property of viewing an object which makes objects appear smaller the further away they are from the viewer. When looking down a long pole the pole appears to be tapered, but our understanding of the real world tells us that this is not so. Thus to add realism to a three dimensional computer display it is often desirable to add perspective to the display, this program is identical to the program THREE DIMENSIONAL SHAPE 1 except that an additional subroutine has been added to remove hidden lines, and the drawing routine has been modified to incorporate the perspective algorithm.

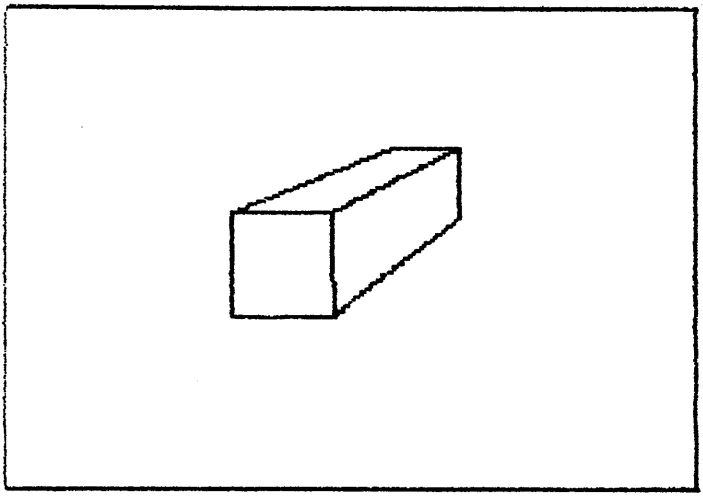
RUNNING THE PROGRAM

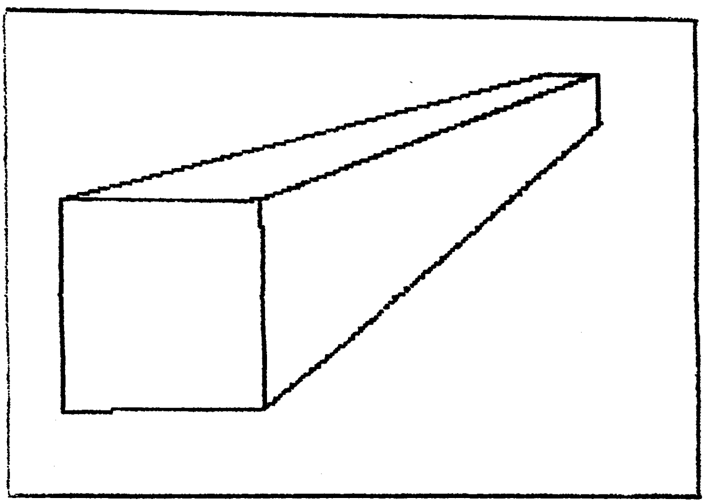
The parameters and data tables required by this program are the same as those used for the program THREE DIMENSIONAL SHAPE 2, consult this program for information.

PROGRAM STRUCTURE

Lines 1 to 5995 are identical to THREE DIMENSIONAL SHAPE 1, (consult for details) except for the following:

2000-2240 shape drawing routine incorporating perspective
algorithm in lines 2030 to 2045
6000-6140 subroutine to check for hidden surfaces





```
   1 REM 3D DRAWING 4
   2 REM ************************
   3 REM
  10 REM a three dimensional sha
pe is drawn by this program
  20 REM the rotation position a
nd scale of the object
  30 REM can be changed to give
different viewing angles.
  40 REM the program incorporate
s a routine to remove hidden lin
es
  50 REM the object is displayed
 with perspective
  55 REM
  60 REM draw border around scre
en
  65 REM
  70 GO SUB 900
  80 REM
  90 REM set up constants variab
les and arrays
  95 REM
 100 DIM a(4,4)
 110 DIM b(4,4)
 115 DIM c(3)
 117 DIM d(3)
 120 LET sx=.1
 130 LET sy=.1
 140 LET sz=.4
 150 LET tx=-20
 160 LET ty=-50
 170 LET tz=1
 180 LET rx=1*3.14159/180
 190 LET ry=1*3.14159/180
 200 LET rz=1*3.14159/180
 400 REM main program loop
 410 GO SUB 1000
 420 GO SUB 5000
 430 GO SUB 3000
 440 GO SUB 4000
 450 GO SUB 6000
 500 STOP
 895 REM
 900 REM border drawing subrouti
ne
 905 REM
 910 PLOT 0,0
 920 DRAW 255,0
 930 DRAW 0,175
 940 DRAW -255,0
 950 DRAW 0,-175
 960 RETURN
 995 REM
1000 REM initialise shape
1005 REM
1010 LET np=8
1020 LET ne=4
1030 LET nf=6
```

```
1040 DIM s(3,np)
1050 DIM e(nf,ne,2)
1060 DIM m(3,np)
1100 REM
1110 FOR n=1 TO np
1120 READ s(1,n),s(2,n),s(3,n)
1130 NEXT n
1140 FOR f=1 TO nf
1150 FOR e=1 TO ne
1160 READ e(f,e,1),e(f,e,2)
1170 NEXT e
1180 NEXT f
1195 REM
1200 REM x,y,z point coordinates
1205 REM
1210 DATA 0,0,200,200,0,200,200,
0,0,0,0,0
1220 DATA 0,200,200,200,200,200,
200,200,0,0,200,0
1295 REM
1300 REM connection data
1305 REM
1310 DATA 1,2,2,3,3,4,4,1
1320 DATA 5,1,1,4,4,8,8,5
1330 DATA 6,5,5,8,8,7,7,6
1340 DATA 2,6,6,7,7,3,3,2
1350 DATA 1,5,5,6,6,2,2,1
1360 DATA 3,7,7,8,8,4,4,3
1900 RETURN
1995 REM
2000 REM draw shape with perspec
tive
2005 REM
2020 FOR e=1 TO ne
2030 LET v1=e(f,e,1)
2033 LET pe=ABS (300/(m(3,v1)-30
0))
2040 LET v2=e(f,e,2)
2043 LET pf=ABS (300/(m(3,v2)-30
0))
2045 IF v1=0 THEN GO TO 2240
2050 LET xb=pe*m(1,v1)
2060 LET yb=pe*m(2,v1)
2070 LET xe=pf*m(1,v2)
2080 LET ye=pf*m(2,v2)
2090 LET ds=1
2100 LET p=xe-xb
2110 LET q=ye-yb
2120 LET r=SQR (p*p+q*q)
2130 LET lx=p/r
2140 LET ly=q/r
2150 FOR i=0 TO r STEP ds
2160 LET x=xb+i*lx
2170 LET y=yb+i*ly
2180 IF x>255 THEN GO TO 2230
2190 IF y>175 THEN GO TO 2230
2200 IF x<0 THEN GO TO 2230
2210 IF y<0 THEN GO TO 2230
2220 PLOT x,y
```

```
2230 NEXT i
2240 NEXT e
2900 RETURN
2995 REM
3000 REM set transformation matr
ix
3005 REM
3010 LET a(1,1)=COS (ry)*COS (rz
)
3020 LET a(1,2)=COS (ry)*SIN (rz
)
3030 LET a(1,3)=-SIN (ry)
3040 LET a(1,4)=0
3050 LET a(2,1)=COS (rx)*(-SIN (
rz))+SIN (rx)*SIN (ry)*COS (rz)
3060 LET a(2,2)=COS (rx)*COS (rz
)+SIN (rx)*SIN (ry)*SIN (rz)
3070 LET a(2,3)=SIN (rx)*COS (ry
)
3080 LET a(2,4)=0
3090 LET a(3,1)=(-SIN (rx))*(-SI
N (rz))+COS (rx)*SIN (ry)*COS (r
z)
3100 LET a(3,2)=-SIN (rx)*COS (r
z)+COS (rz)*SIN (ry)*SIN (rz)
3110 LET a(3,3)=COS (rx)*COS (ry
)
3120 LET a(3,4)=0
3130 LET a(4,1)=0
3140 LET a(4,2)=0
3150 LET a(4,3)=0
3160 LET a(4,4)=1
3195 REM
3200 REM set up scaling and tran
slation matrix
3205 REM
3210 LET b(1,1)=sx*a(1,1)
3220 LET b(1,2)=sx*a(1,2)
3230 LET b(1,3)=sx*a(1,3)
3240 REM
3250 LET b(2,1)=sy*a(2,1)
3260 LET b(2,2)=sy*a(2,2)
3270 LET b(2,3)=sy*a(2,3)
3280 REM
3290 LET b(3,1)=sz*a(3,1)
3300 LET b(3,2)=sz*a(3,2)
3310 LET b(3,3)=sz*a(3,3)
3320 REM
3330 LET b(4,1)=tx
3340 LET b(4,2)=ty
3350 LET b(4,3)=tz
3900 RETURN
3995 REM
4000 REM perform translation
4005 REM
4010 FOR q=1 TO np
4015 REM
4020 LET xt=s(1,q)-xc
4030 LET yt=s(2,q)-yc
```

```
4040 LET zt=s(3,q)-zc
4045 REM
4050 LET m(1,q)=xc+(xt*b(1,1)+yt*b(2,1)+zt*b(3,1)+b(4,1))
4060 LET m(2,q)=yc+(xt*b(1,2)+yt*b(2,2)+zt*b(3,2)+b(4,2))
4070 LET m(3,q)=zc+(xt*b(1,3)+yt*b(2,3)+zt*b(3,3)+b(4,3))
4080 NEXT q
4900 RETURN
4995 REM
5000 REM find centroid
5005 REM
5010 LET p=0: LET q=0: LET r=0
5020 FOR i=1 TO np
5030 LET p=p+s(1,i)
5040 LET q=q+s(2,i)
5050 LET r=r+s(3,i)
5060 NEXT i
5070 LET xc=p/np
5080 LET yc=q/np
5090 LET zc=r/np
5900 RETURN
5995 REM
6000 REM hidden surface check
6005 REM
6010 FOR f=1 TO nf
6020 FOR j=1 TO 3
6030 LET c(j)=m(j,e(f,1,2))-m(j,e(f,1,1))
6040 LET d(j)=m(j,e(f,2,1))-m(j,e(f,2,2))
6050 NEXT j
6060 LET p1=c(2)*d(3)-c(3)*d(2)
6070 LET p2=c(3)*d(1)-c(1)*d(3)
6080 LET p3=c(1)*d(2)-c(2)*d(1)
6090 LET q1=-120-m(1,e(f,1,2))
6100 LET q2=-80-m(2,e(f,1,2))
6110 LET q3=500-m(3,e(f,1,2))
6120 LET w=p1*q1+p2*q2+p3*q3
6130 IF w>=0 THEN GO SUB 2000
6140 NEXT f
6900 RETURN
```

INDEX

TIMEX SINCLAIR

1064-8

Color Graphics

Nick Hampshire

A dazzling display of color graphics in 45 complete programs for the Timex Sinclair 2048 and 2068 microcomputers. The author clearly explains the theory behind high-resolution-graphics plotting by presenting practical, concrete programming examples. Applications of these graphics displays range from art to games to educational simulations in math, science, and business. All program listings have been tested and are annotated for easy reference and modification. Use them as is or change them to suit your own specific needs!

Programs include color plotting; drawing maps, rainbows, geometric figures, pie charts, and bar and line graphs; character and shape design; and moving figures on the screen. The programs build to reveal the techniques of three-dimensional drawing, including adding perspective, shading, and color to achieve stunning results in high-resolution graphics.

TIMEX SINCLAIR™ BASIC

Joseph Charles

Intended for beginning users of Timex Sinclair microcomputers, this book takes you from the first steps of programming to writing your own programs for business, engineering, household and games applications, and more.

Learn how to edit programs; program with the use of number and character strings, functions, and subroutines; and learn how to modify programs written for other computers so that they will run on your computer.

Timex Sinclair BASIC includes scores of actual program examples to help you understand and apply the concepts, various exercises to help you learn and gain confidence on the keyboard, and solutions to selected exercises so that you can check your overall progress on the computer. #2101-4, 192 pages.

HAYDEN BOOK COMPANY, INC.
Rochelle Park, New Jersey

ISBN 0-8104-1064-8